НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ
Кафедра автоматизованих систем обробки інформації і управління

До захисту допущено:
В.о. завідувача кафедри
_____ *Олександр ПАВЛОВ*
(підпис) (вл.ім'я, прізвище)
"___"_____2020 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інформаційні управляючі системи та технології»
спеціальності 122 «Комп'ютерні науки та інформаційні технології»

на тему: «*Система інформаційної підтримки навчальних курсів*»

Виконав: студент *IV* курсу, групи *ІС-62*

Мусієнко Віталій Сергійович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник *ст. вик. Ковтунець Олесь Володимирович* _____
(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові) (підпис)

Консультант з графічної документації *доц., к.т.н. Новінський Валерій Петрович* _____
(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові) (підпис)

Рецензент *доц., к.т.н. Пасько Віктор Петрович* _____
(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові) (підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2020 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет (інститут) *інформатики та обчислювальної техніки*
(повна назва)

Кафедра *автоматизованих систем обробки інформації і управління*
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 122 «Комп'ютерні науки та інформаційні технології»

Освітньо-професійна програма «Інформаційні управляючі системи та технології»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ *Олександр ПАВЛОВ*
(підпис) (вл.ім’я, прізвище)
“___”_____2020 р.

ЗАВДАННЯ
на дипломний проєкт студенту

Мусієнку Віталію Сергійовичу
(прізвище, ім’я, по батькові)

1. Тема проєкту «*Система інформаційної підтримки навчальних курсів*»

керівник проєкту *Ковтунець Олесь Володимирович, ст. вик.*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від “*7*”*травня* 2020 р. №1081-с
2. Термін подання студентом проєкту “*01*”*червня* 2020 року
3. Вихідні дані до проєкту
Технічне завдання

4. Зміст пояснювальної записки
1. Загальні положення: основні визначення та терміни, опис предметного середовища, огляд ринку програмних продуктів, постановка задачі
2. Інформаційне забезпечення: вхідні дані, вихідні дані, опис структури бази даних
3. Математичне забезпечення: змістовна та математична постановки задачі, обґрунтування та опис методу розв’язання
4. Програмне та технічне забезпечення: засоби розробки, вимоги до технічного забезпечення, архітектура програмного забезпечення, побудова звітів

5. Технологічний розділ: керівництво користувача, методика випробувань програмного продукту

5. Перелік графічного матеріалу

1. Схема структурна варіантів використань

3. Схема бази даних

4. Схема структурна класів програмного забезпечення

5. Схема структурна послідовності

6. Схема структурна компонентів програмного забезпечення

7. Креслення екранних форм

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання *«13» квітня 2020 року*

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	*Вивчення рекомендованої літератури*	10.03.2020 р.	
2.	*Аналіз існуючих методів розв'язання задачі*	15.03.2020 р.	
3.	*Постановка та формалізація задачі*	28.03.2020 р.	
4.	*Розробка інформаційного забезпечення*	7.05.2020 р.	
5.	*Алгоритмізація задачі*	22.03.2020 р.	
6.	*Обґрунтування використовуваних технічних засобів*	30.03.2020 р.	
7.	*Розробка програмного забезпечення*	10.04.2020 р.	
8.	*Налагодження програми*	17.04.2020 р.	
9.	*Виконання графічних документів*	23.04.2020 р.	
10.	*Оформлення пояснювальної записки*	01.05.2020 р.	
11.	*Подання ДП на попередній захист*	15.05.2020 р.	
12.	*Подання ДП на основний захист*	01.06.2020 р.	
13.	*Подання ДП рецензенту*	02.06.2020 р.	

Студент Віталій МУСІЄНКО

Керівник Олесь КОВТУНЕЦЬ

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	ДП 6213.00.000 ПЗ	Пояснювальна записка	94	
3	А4	ДП 6213.01.000 ТЗ	Технічне завдання	9	
4	А3	ДП 6213.02.000 ССВ	Схема структурна варіантів використань	1	
5	А3	ДП 6213.03.000 СБД	Схема бази даних	1	
6	А3	ДП 6213.04.000 ССК	Схема структурна класів програмного	1	
			забезпечення		
7	А3	ДП 6213.05.000 ССП	Схема структурна послідовності	1	
8	А3	ДП 6213.06.000 ССК	Схема структурна компонентів	1	
9	А3	ДП 6213.07.000 КЕ	Креслення вигляду екранних форм	1	

Зм.	Арк.	ПІБ	Підп.	Дата	ДП 6213 00.000.00		
Розробн.		Мусієнко В.С.			Відомість дипломного проєкту	Літ.	Лист 1 / Листів 1
Керівн.		Ковтунець О.В.					
Консульт.						КПІ ім. Ігоря Сікорського Каф. АСОІУ Гр. ІС-62	
Н/контр.		Новінський В.П.					
В.о.зав.каф.		Павлов О.А.					

Пояснювальна записка

до дипломного проєкту

на тему: Система інформаційної підтримки навчальних курсів

Київ – 2020 року

АНОТАЦІЯ

Структура та обсяг роботи. Пояснювальна записка дипломного проєкту складається з шести розділів, містить 14 рисунків, 54 таблиці, 1 додаток, 18 джерел.

Дипломний проєкт присвячений розробці системи інформаційної підтримки навчальних курсів. Цілями створення даної системи є допомога в організації навчального процесу на курсах, надання можливості швидко і зручно знаходити курси по власних інтересах та покращення ефективності навчання студентів. Для досягнення поставлених цілей необхідно розв’язати наступні задачі:

- Розробити зручний спосіб комунікації учасників всередині системи;
- Розробити систему рекомендацій на основі інтересів користувача;
- Розробити зручний інтерфейс для надання користувачу тільки необхідної інформації.

У розділі інформаційного забезпечення описано вхідні та вихідні дані, продемонстрована схема структурна бази даних та надана її характеристика.

Розділ математичного забезпечення присвячений опису математичної задачі, яку необхідно розв’язати. Обґрунтований та описаний метод розв’язання. Наведено висновок до розділу.

Програмне забезпечення, засоби розробки, вимоги до технічного забезпечення, архітектура програмного забезпечення описані в четвертому розділі.

У технологічному розділі описані керівництво користувача, випробування програмного продукту та їх результати.

КЛЮЧОВІ СЛОВА: НАВЧАЛЬНІ КУРСИ, СИСТЕМА, КОЛАБОРАТИВНА ФІЛЬТРАЦІЯ, РЕКОМЕНДАЦІЇ, JAVA

ABSTRACT

Structure and scope of work. The explanatory note of the bachelor's thesis consists of six sections, contains 14 drawings, 54 tables, 1 application, 18 sources.

The bachelor's thesis is devoted to the development of an information support system for training courses. The goals of this system are to help organize the learning process in the courses, to provide opportunities to quickly and easily find courses in their own interests and to improve the student learning effectiveness. To achieve these goals it is necessary to solve the following tasks:

- Develop a convenient way for participants to communicate within the system

- Develop a system of personalized recommendations based on user interests

- Develop a user-friendly interface to provide the user with only the necessary information

The section of information support describes the input and output data, demonstrates the structural structure of the database and provides its characteristics.

The section on mathematical software is devoted to the description of the mathematical problem to be solved. The method of solution is substantiated and described. The conclusion to the section is given.

Software, development tools, hardware requirements, software architecture are described in the fourth section.

The technology section describes the user manual, software product tests and their results.

KEYWORDS: TRAINING COURSES, SYSTEM, COLLABORATIVE FILTRATION, RECOMMENDATIONS, JAVA

ЗМІСТ

ВСТУП

Ми живемо в такий час, коли ледь не кожного дня з’являються якісь нові галузі, напрямки та професії. Це значить, що попит на різноманітних спеціалістів також зростає, а отже – зростає необхідність вивчати щось нове. Безумовно, майже кожен із нас має вільний доступ до інтернету, книжок, але далеко не кожен може самотужки повноцінно розібратися в питанні, яке його цікавить. Тому, допоки вчені не винайдуть спосіб завантажувати знання людям у голову, як на флеш-накопичувач, необхідність у викладачах нікуди не зникне.

Поява онлайн-освіти якщо не спровокувала, то багато в чому значно сприяла розвитку концепції безперервного навчання, яка з кожним роком знаходить все більше нових прихильників. Завдяки онлайну, вам уже не потрібно витрачати години, місяці або роки свого життя, щоб отримати ті чи інші навички або знання. Онлайн-формати дозволили значно розширити спектр предметів, доступних для вивчення, зміст курсів став більш ємним і структурованим, а процес навчання - більш комфортним.

Різноманітних курсів у наш час дуже багато. Це одночасно є і перевагою, і недоліком. Адже у всій інформації, яка нас оточує, можна легко загубитись і не знайти те, що тобі потрібно. Тому всю цю інформацію необхідно структурувати і подати у зручному вигляді, щоб усі бажаючі могли в ній легко орієнтуватися.

1. ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1 Опис предметного середовища

Нова інформація з’являється так само швидко, як і стає не актуальною. Тому потрібно постійно вивчати щось нове і самовдосконалюватись, щоб іти в ногу із часом. Але що робити, якщо школа дає тільки базові знання, а програма університету не відповідає тому, що актуальне на ринку прямо зараз? Вирішити цю проблему допомагають навчальні курси.

Одним із найважливіших факторів у навчальному процесі є його правильна організація. В наш час недостатньо просто надати інформацію, кількість якої невпинно зростає кожного дня, її необхідно структурувати та грамотно подати. Різноманітної інформації зараз стільки, що іноді буває досить важко не заплутатися в ній і знайти саме те, що тобі потрібно. Кількість навчальних курсів, напевно, неможливо навіть порахувати, тому шукати їх самотужки іноді буває досить важко.

Як часто у вас бували ситуації, коли ви забували про якесь заняття, не могли знайти контакт викладача, щоб задати йому кілька додаткових питань, не знали, яка аудиторія вам потрібна? На мою думку, це досить поширені проблеми. Під час навчання у студентів завжди виникають питання, і дуже важливо швидко і доступно дати їм відповідь.

Поставлене мною завдання – розробити систему, яка б вирішувала перераховані вище проблеми. Програмний продукт повинен допомагати студентам, викладачам та менеджерам правильно організувати навчальний процес та надавати їм можливість спілкуватись, обмінюватись необхідною інформацією в межах одного середовища. Також повинна бути розроблена система рекомендацій, яка буде пропонувати студентам нові курси, опираючись на їх інтереси та інтереси інших користувачів.

1.1.1 Опис процесу діяльності

Об’єктом автоматизації є процес навчання на курсах. Розглянемо дії, які може виконувати користувач під час використання даної системи. На рисунку 1.1 зображена діаграма діяльності для студента.

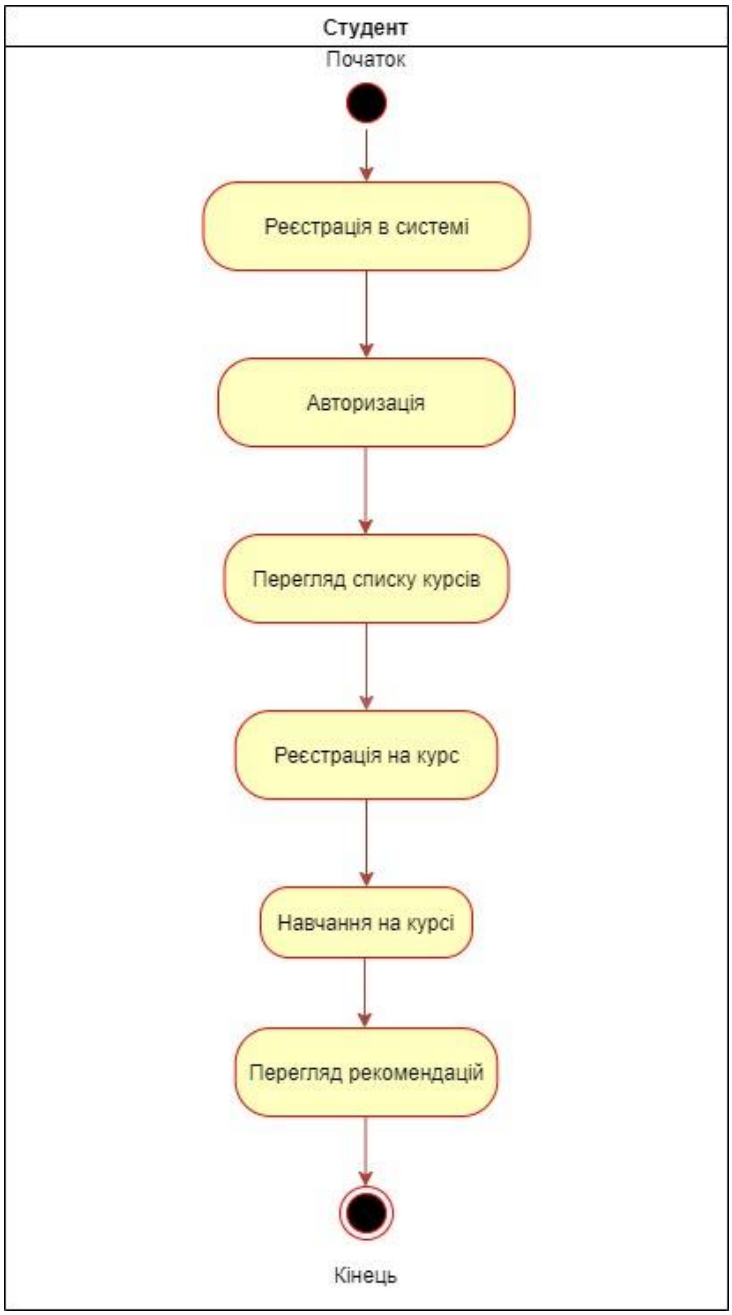


Рисунок 1.1 – Діаграма діяльності студента

1.1.2 Опис функціональної моделі

Система інформаційної підтримки навчальних курсів представляє із себе веб-застосунок, для користування яким необхідно зареєструватися. Схема структурна варіантів використання наведена у графічному матеріалі. Система передбачає 4 типи користувачів, які мають різні можливості та рівні доступу. Детальніше вони описані нижче.

1) Актори, які будуть взаємодіяти в межах розробленого продукту:

- STUDENT – студент;
- MANAGER – менеджер, який закріплюється за кожним студентом для моніторингу навчального процесу;
- TEACHER – викладач курсів;
- ADMIN – адміністратор системи.

2) Функції, які будуть виконуватись акторами

STUDENT:

- Реєстрація в системі;
- Реєстрація на курс;
- Перегляд усіх курсів;
- Перегляд розкладу занять;
- Чат з іншими учасниками системи;
- Перегляд рекомендованих курсів;
- Перегляд профілю менеджера та викладачів.

MANAGER:

- Перегляд списку своїх студентів;
- Перегляд профілю студента;
- Перегляд відвідуваність своїх студентів;
- Створення групових чатів.

TEACHER:

- Перегляд усіх своїх груп та курсів;
- Перегляд профілю своїх студентів;
- Перегляд профілю менеджера своїх студентів;
- Створення, редагування та видалення занять;
- Перевірка присутності студентів на заняттях.

ADMIN:

- Створення та редагування профілів менеджера та викладача;
- Перегляд профілю усіх учасників системи;
- Створення та редагування курсів і груп;
- Перегляд списку курсів;
- Перегляд списку груп;
- Перегляд відвідуваності по групах.

1.1.3 Огляд наявних аналогів

У наш час існує багато платформ, які надають доступ до онлайн курсів, таких як Coursera [1], Prometeus, Skillbox тощо. Безумовно, дуже зручно отримувати нові знання, не виходячи з дому. Також дистанційне навчання може бути не тільки корисним, а і єдиним можливим варіантом в деяких нестандартних ситуаціях. Однак далеко не всі галузі можна опанувати, сидячи перед монітором комп’ютера. Деякі напрямки вимагають комунікації між людьми, постійного живого спілкування, тому, на мою думку, офлайн курси все одно будуть залишатись актуальними.

Одним із основних аналогів можна назвати систему moodle. Moodle – це одна із найпоширеніших систем керування курсами, також відома як система керуванням навчанням або віртуальне навчальне середовище [2, 3].

Ця система надає великий набір інструментів для підтримки та проведення навчального процесу, але має і свої недоліки, такі як:

- Складність системи;
- Відсутність професійної технічної підтримки;
- Необхідність збирати систему з нуля;
- Потребує технічних компетенцій в області веб-розробки від викладача;
- Відсутність системи рекомендацій.

Сьогодні існує велика кількість різноманітних курсів на будь-який смак. Але ідея для даного сервісу виникла через те, що всі курси доводиться шукати окремо. Інформації в інтернеті неймовірно багато, тому її необхідно якось фільтрувати. Безумовно, існують платформи, які пропонують перелік різних курсів в одному місці. Наприклад, існують такі школи англійської мови, як Green Forest, Project 12. Вони надають користувачу доступ до навчальних матеріалів та важливої інформації в особистому кабінеті, але їх недоліком є те, що кожен сервіс пропонує тільки власні курси. А ідея даної роботи в тому, щоб зібрати пропозиції від різних людей, компаній, брендів в одному місці і створити універсальний особистий кабінет для студента, позбавивши його необхідності власноруч шукати необхідну інформацію.

1.2 Постановка задачі

1.2.1 Призначення розробки

Призначенням даної розробки є допомога в організації навчального процесу на курсах. Система повинна давати можливість студентам, викладачам та менеджерам зібрати всю необхідну для навчального процесу інформацію в одному місці.

1.2.2 Цілі та задачі розробки

Цілі створення системи:

- Допомога в організації навчального процесу на курсах;
- Можливість швидко і зручно знаходити курси по власних інтересах;
- Покращити ефективність навчання студентів.

Для досягнення поставлених цілей необхідно розв’язати наступні задачі:

- Розробити зручний спосіб комунікації учасників всередині системи;
- Розробити систему персоналізованих рекомендацій на основі інтересів користувача;
- Розробити зручний інтерфейс для надання користувачу тільки необхідної інформації;
- Розробити просту та зрозумілу систему контролю відвідуваності студентів.

ВИСНОВКИ ДО РОЗДІЛУ

На даному етапі були описані мета та цілі розробки, предметне середовище, наведені існуючі аналоги, проведена їхня порівняльна характеристика, визначено переваги та недоліки. Також була дана коротка характеристика учасників системи та їх можливостей, створені основні вимоги до системи, які необхідно реалізувати, побудована діаграма діяльності.

2 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Вхідні дані

Вхідні дані надходять в систему від декількох акторів, а саме:

- Студенти;
- Адміністратор;
- Менеджер;
- Викладач;

Дані вводяться в систему за допомогою інтерфейсу веб-застосунку. Далі будуть розглянуті детальніше вхідні дані від різних користувачів системи:

- Дані, які надходять від адміністратора та користувачів:
  - Реєстрація:
    1. Ім’я;
    2. Прізвище;
    3. Електронна адреса;
    4. Пароль.
- Дані, які надходять від усіх користувачів:
  - Пошук курсу.
    1. Назва курсу або ключові слова
- Дані, які надходять від адміністратора:
  - Створення курсу:
    1. Назва курсу;
    2. Опис курсу;
    3. Викладач курсу;
    4. Дата початку;
    5. Дата закінчення;
    6. Статус.
  - Пошук викладача або менеджера

1. Ім’я або прізвище викладача/менеджера

- Створення групи:
  1. Назва групи;
  2. Курс, до якого належить група.
- Створення групового чату:
  1. Назва чату;
  2. Група, за якою закріплений чат.
- Реєстрація облікового запису для менеджера;
- Реєстрація облікового запису для викладача;
- Створення занять:
  1. Назва заняття;
  2. Тривалість заняття;
  3. Дата заняття;
  4. Курс, до якого належить заняття.

- Дані, які надходять від викладача:
  - Відвідуваність студентів:
    1. Присутність/відсутність студентів на конкретних заняттях;
    2. Причина відсутності.
  - Створення розкладу занять:
    1. Ідентифікатор заняття;
    2. Дата заняття.
    3. Курс, до якого належить заняття.

2.2 Вихідні дані

1. Вихідні дані, які формуються при створення курсу:
   - Назва курсу;
   - Статус;
   - Викладач.
2. Вихідні дані, які формуються при створенні групи:

- Назва групи;
- Кількість студентів;
- Курс.

3. Вихідні дані, які формуються при створенні групового чату:
   - Назва чату;
   - Група, за якою він закріплений.
4. Вихідні дані, які формуються при створенні заняття:
   - Назва заняття;
   - Курс, за яким закріплене заняття.
5. Вихідні дані, які формуються при пошуку курсу:
   - Список курсів, що задовольняють введеній назві.
6. Вихідні дані, які формуються при пошуку менеджера або викладача:
   - Список користувачів, які задовольняють введеному імені або прізвищу.

2.3 Опис структури бази даних

У даній дипломній роботі в якості бази даних була використана MongoDB. У ній застосовується безсхемна організація неструктурованих або слабкоструктурованих даних, тобто організація, яка не потребує опису схеми бази даних. Такий спосіб організації даних не призначений для того, щоб пов’язувати документи різних колекцій через ключові поля. В MongoDB немає власних засобів для створення відношень між колекціями і документами.

Документоорієнтована СКБД MongoDB призначена для зберігання даних в документах колекцій у форматі JSON, які не мають чіткої структури і призначені для зберігання будь-якого типу даних. Такий формат і організація зберігання даних в СКБД MongoDB забезпечує оперативну оброку даних, їх запис та вивід результатів [4].

Проте для наглядності та простоти розуміння була побудована ER-діаграма та описані колекції в контексті СКБД MongoDB. ER-діаграма бази даних зображена у графічному матеріалі.

Детальніше сутності описані в таблицях 2.1 – 2.11.

Таблиця 2.1 – Колекція користувачів

Назва колекції	**Назва поля**	**Тип даних**	**Детальна інформація**
User	Id	ObjectId	Ідентифікатор користувача в базі даних
	firstName	String	Ім’я користувача
	lastName	String	Прізвище користувача
	managerId	String	Ідентифікатор менеджера користувача
	Role	String	Роль користувача в системі
	email	String	Електронна адреса користувача
	password	String	Пароль користувача для входу в систему

Таблиця 2.2 – Колекція курсів

Назва сутності	**Назва поля**	**Тип даних**	**Детальна інформація**
Course	id	String	Id курсу в базі даних
	Name	String	Назва курсу
	Description	String	Опис курсу
	teacherId	String	Ідентифікатор викладача курсу
	status	String	Статус курсу (відкритий, завершений, активний)

Таблиця 2.3 – Колекція груп

Назва сутності	**Назва поля**	**Тип даних**	**Детальна інформація**
Group	Name	String	Назва групи
	courseId	String	Ідентифікатор курсу в базі даних, за яким закріплена група
	Capacity	int	Кількість студентів у групі
	Status	String	Статус групи (відкрита, закрита)

Таблиця 2.4 – Колекція занять

Назва сутності	**Назва поля**	**Тип даних**	**Детальна інформація**
Lesson	id	String	Ідентифікатор заняття в базі даних
	Name	String	Назва заняття
	courseId	String	Ідентифікатор курсу в базі даних, до якого відноситься дане заняття

Таблиця 2.5 – Колекція студентів та груп

Назва сутності	**Назва поля**	**Тип даних**	**Детальна інформація**
UserGroup	id	String	Ідентифікатор сутності в базі даних
	userId	String	Ідентифікатор студента в базі даних
	groupId	String	Ідентифікатор групи в базі даних

Таблиця 2.6 – Колекція рекомендацій

Назва сутності	**Назва поля**	**Тип даних**	**Детальна інформація**
Recommendation	id	String	Ідентифікатор сутності в базі даних
	courseId	String	Ідентифікатор курсу
	recommendCourseId	String	Ідентифікатор курсу, який рекомендований для обраного курсу

Таблиця 2.7 – Колекція розкладу занять

Назва сутності	**Назва поля**	**Тип даних**	**Детальна інформація**
Schedule	id	String	Ідентифікатор сутності в базі даних
	groupId	String	Ідентифікатор групи
	date	String	Дата заняття
	lessonId	String	Ідентифікатор заняття

Таблиця 2.8 – Колекція розкладу занять

Назва сутності	**Назва поля**	**Тип даних**	**Детальна інформація**
Attendance	id	String	Ідентифікатор сутності в базі даних
	userId	String	Ідентифікатор студента
	attendanceType	String	Тип присутності на занятті (присутній, відсутній, відсутній по хворобі тощо)
	scheduleId	String	Ідентифікатор розкладу

Таблиця 2.9 – Колекція чатів

Назва сутності	**Назва поля**	**Тип даних**	**Детальна інформація**
Chat	id	String	Ідентифікатор сутності в базі даних
	chatName	String	Назва чату
	groupId	String	Назва групи, за якою закріплений чат

Таблиця 2.10 – Колекція повідомлень

Назва сутності	**Назва поля**	**Тип даних**	**Детальна інформація**
Message	id	String	Ідентифікатор сутності в базі даних
	senderId	String	Назва чату
	date	Date	Назва групи, за якою закріплений чат
	message	String	Повідомлення
	chatId	String	Ідентифікатор чату

Таблиця 2.11 – Колекція особистих чатів користувачі

Назва сутності	**Назва поля**	**Тип даних**	**Детальна інформація**
UserChat	id	String	Ідентифікатор сутності в базі даних
	senderId	String	Ідентифікатор першого користувача
	receiverId	Date	Ідентифікатор другого користувача
	chatId	String	Ідентифікатор чату

ВИСНОВОК ДО РОЗДІЛУ

У даному розділі були описані вхідні дані, які система приймає від користувачів (адміністратора, менеджерів, викладачів та студентів) і вихідні дані, які є результатом обробки вхідних даних. Описана структура бази даних, надано пояснення до кожної сутності.

3 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Змістовна постановка задачі

Система інформаційної підтримки навчальних курсів передбачає можливість рекомендувати студентам курси, подібні до тих, на які вони вже зареєстровані. Нехай маємо **n** зареєстрованих в системі користувачів та **m** опублікованих курсів. Переглянути всі можливі курси може бути досить складно, якщо їх дуже багато. Відповідно, щоб знайти курс, який може зацікавити користувача, йому може знадобитися велика кількість часу.

Також цілком логічно припустити, що якщо користувач зареєструвався на один курс, то його може зацікавити інший, який схожий на перший. Тому необхідно згрупувати курси, які подібні між собою, та надати їх користувачу у вигляді рекомендацій [5].

3.2 Математична постановка задачі

Вхідні дані:

- $U = \{u_1, u_2, \ldots, u_n\}$ – множина зареєстрованих користувачів, де n – кількість користувачів у системі;
- $R = \{r_1, r_2, \ldots, r_m\}$ – множина опублікованих курсів, де m – кількість курсів у системі;
- F – матриця реєстрацій на курс, де f_{ur} = [користувач u зареєструвався на курс r] [6].

Змінні:

- $\overrightarrow{Ri}$ – вектор реєстрацій на курс r_i;
- S_{ij} – числова характеристика подібності і-го курсу до j-го ;
- $x_i = \begin{cases} 1 \\ 0 \end{cases}$, курс r_i рекомендований або ні.

Обмеження:

- $S_{ij} \geq 0.5$ (3.1)
- Числову характеристику подібності курсів обраховуємо через формулу косинусної міри [7]:

$$S_{ij} = cos(\overrightarrow{R_i}, \overrightarrow{R_j}) = \frac{\overrightarrow{R_i} \cdot \overrightarrow{R_j}}{\|\overrightarrow{R_i}\| \cdot \|\overrightarrow{R_j}\|} = \frac{\sum_{k=1}^{n} R_{ik} R_{jk}}{\sqrt{\sum_{k=1}^{n} {R_{ik}}^2} \sqrt{\sum_{k=1}^{n} {R_{jk}}^2}} \quad (3.2)$$

Вихідні дані:

- R(r_0) – список курсів, подібних до r_0.

3.3 Обґрунтування методу розв'язання

Навчання на курсах може тривати досить довго, іноді цей процес може займати до кількох місяців. Відповідно користувач зможе оцінити курс тільки після його завершення, а деякі користувачі навіть не ставлять оцінки. Тому використання алгоритму рекомендацій, який базується на оцінках користувачів, не є найкращим рішенням в даному випадку, адже такий алгоритм буде видавати правдиві результати лише через якийсь проміжок часу (після проходження багатьох курсів). Система навіть може не видавати користувачу рекомендацій весь час, протягом якого він користується даним ресурсом. Для вирішення цієї проблеми було використано алгоритм колаборативної фільтрації Item-to-Item, який базується на двійкових даних (реєструвався користувач на курс чи ні) [8, 9]. Він розраховує подібність курсів як косинус між векторами реєстрацій в матриці користувачів і курсів.

3.4 Опис методів розв'язання

Основною математичною задачею, яку розв'язує даний алгоритм, є задача пошуку схожих між собою курсів.

Нехай нам відома таблиця реєстрацій по кожному користувачу та курсу.

Таблиця 3.1 – Таблиця реєстрацій.

	R_1	R_2	R_3	…	R_m
U_1	F_{11}	F_{12}	F_{13}	…	F_{1m}
U_1	F_{21}	F_{22}	F_{23}	…	F_{2m}
U_3	F_{31}	F_{32}	F_{33}	…	F_{3m}
…	…	…	…	…	…
U_n	F_{n1}	F_{n2}	F_{n3}	…	F_{nm}

Потрібно знайти список рекомендацій для кожного з курсів. Для цього потрібно побудувати вектори реєстрацій для кожного курсу та знайти косинус між ними. Вектор для кожного курсу – це стовпець R.

Далі знаходимо числову характеристику подібності кожної пари курсів за формулою (3.1).

Таблиця 3.2 – Таблиця подібностей курсів.

	R_1	R_2	R_3	…	R_m
R_1	S_{11}	S_{12}	S_{13}	…	S_{1m}
R_1	S_{21}	S_{22}	S_{23}	…	S_{2m}
R_3	S_{31}	S_{32}	S_{33}	…	S_{3m}
…	…	…	…	…	…
R_m	S_{m1}	S_{m2}	S_{m3}	…	S_{mm}

Причому для кожного S_{ij} виконується умова: $0 \leq S_{ij} \leq 1$, $i = \overrightarrow{1, n}$; $j = \overrightarrow{1, n}$.

Якщо виконується умова (3.1), значить курси достатньо подібні між собою і ми приймаємо відповідну змінну *x* за одиницю.

Описаний вище метод можна представити у вигляді покрокового алгоритму

1) Знаходимо вектор реєстрацій для кожного курсу;

2) Знаходимо коефіцієнт подібності для кожної пари курсів за формулою (3.2);

3) Якщо обмеження 3.1 виконується, значить відповідна змінна x рівна одиниці;

4) Вибрати всі курси, для яких x = 1.

Розглянемо приклад. Нехай маємо таблицю реєстрацій:

Користувач	Курс 1	Курс 2	Курс 3
Василь	Зареєстрований	Не зареєстрований	Зареєстрований
Петро	Не зареєстрований	Зареєстрований	Зареєстрований
Олег	Не зареєстрований	Зареєстрований	Не зареєстрований

В умовах нашої задачі вона матиме вигляд:

	R_1	R_2	R_3
U_1	1	0	1
U_2	0	1	1
U_3	0	1	0

В цьому випадку коефіцієнт подібності між R_1 і R_2 розраховується наступним чином:

$$s_{12} = \frac{(1,0,0)\cdot(0,1,1)}{\|(1,0,0)\| * \|(0,1,1)\|} = 0$$

Між R_1 і R_3:

$$s_{13} = \frac{(1,0,0)\cdot(1,1,0)}{\|(1,0,0)\| * \|(1,1,0)\|} = \frac{1}{\sqrt{2}} \approx 0.71$$

Між R_2 і R_3:

$$s_{23} = \frac{(0,1,1)\cdot(1,1,0)}{\|(0,1,1)\| * \|(1,1,0)\|} = \frac{1}{2} = 0.5$$

Таким чином матриця подібностей набуває вигляду:

	R_1	R_2	R_3
R_1	0	0	1
R_2	0	0	1
R_3	1	1	0

Отже, для курсу R_1 ми можемо рекомендувати R_3, для курсу R_2 можемо рекомендувати курс R_3, а для курсу R_3 можемо рекомендувати R_1 та R_2 [10].

ВИСНОВОК ДО РОЗДІЛУ

У даному розділі описана змістовна постановка задачі та побудована її математична постановка. Також були визначені можливі рішення поставленої задачі, вибрано та обґрунтовано метод, який реалізовано в даному дипломному проєкті (метод колаборативної фільтрації на основі подібності елементів), з приведенням покрокового алгоритму та прикладу розв’язання задачі.

4 ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

4.1 Засоби розробки

Під час створення програмного продукту були використані наступні технології:

- Java;
- Spring Framework;
- Spring boot;
- HTML, CSS, JavaScript;
- Bootstrap;
- Websocket;
- Thymeleaf;
- MongoDB.

Java - це об'єктно-орієнтована мова програмування, розроблена компанією «Sun Microsystems» як основний компонент платформи Java. 2009 року компанія «Oracle», викупила «Sun Microsystems» та продовжила займатися мовою Java. Програми, написані на цій мові, компілюються у байт-код, який при виконанні інтерпретується віртуальною машиною для конкретної платформи [11].

Spring Framework (або коротко Spring) — це універсальний фреймворк з відкритим кодом для Java-платформи. Він забезпечує рішення багатьох задач, з якими зустрічаються Java-розробники і організації, які розробляють інформаційні системи на основі платформи Java [12].

Spring Boot – це розширення фреймворку Spring, яке використовується для створення автономних застосунків. Для того, щоб розгорнути типовий застосунок на основі Spring Framework, необхідно написати певну кількість коду. Причому зазвичай у стандартних проектів цей конфігураційний код є однаковим. Для того, щоб спростити життя розробникам і новачкам, яким на початку важко налаштувати проект, з’явився Spring Boot. Він має заздалегідь

підготовлені частини конфігурації, які активуються залежно від різних умов [13].

HTML, CSS, JavaScript – базовий набір інструментів для веб-розробки. HTML – це стандартизована мова розмітки документів. CSS – формальна мова стилю сторінок, яка використовується для опису їхнього зовнішнього вигляду. JavaScript — це динамічна об'єктно-орієнтована мова програмування. Найчастіше вона використовується для створення сценаріїв веб-сторінок, що надає можливість на стороні клієнта взаємодіяти з користувачем, керувати браузером, обмінюватися даними із сервером та змінювати зовнішній вигляд веб-сторінки [14].

Bootstrap - це набір інструментів з відкритим кодом, який призначений для розробки веб-сайтів та веб-додатків. Він містить шаблони CSS та HTML для форм, кнопок, навігації та інших компонентів інтерфейсу, а також додаткові розширення JavaScript. Даний набір інструментів спрощує та прискорює веб-розробку [15].

Websocket – це протокол, що призначений для обміну інформацією між браузером та веб-сервером в режимі реального часу. Він забезпечує двонаправлений канал зв'язку через один TCP-сокет. WebSocket спроектовано для втілення у веб-браузерах та веб-серверах, але може також використовуватись будь-яким клієнт-серверним застосунком. У веб-застосунках доцільно використовувати протокол за необхідності відображення інформації в реальному часі [16]. В даній роботі Websocket використаний для обміну повідомленнями користувачів всередині системи.

Thymeleaf – це сучасний серверний механізм Java-шаблонів. Основною задачею Thymeleaf є створення елегантного та зручного способу шаблонізації. Він базується на концепції Natural Templates, щоб впровадити свою логіку в файли шаблонів таким чином, щоб цей шаблон не впливав на відображення прототипу дизайну. Це покращує комунікацію в команді і зменшує розрив між дизайнерами та програмістами [17].

MongoDB — це документо-орієнтована система керування базами даних, яка не потребує опису схеми таблиць. Вона є досить гнучкою для формування запитів, дозволяє ефективно зберігати бінарні дані, має повну підтримку індексів, підтримує реплікацію і побудову відмовостійких конфігурацій [18].

У поєднанні з інструментами мови програмування Java та фреймворку Spring, MongoDB є хорошим вибором для розробки веб-застосунків. Оскільки разом ці технології виконують частину роботи замість програміста, автоматично генерують деякі необхідні класи та надають розробнику можливість швидко та зручно керувати даними.

4.2 Вимоги до технічного забезпечення

4.2.1 Загальні вимоги

Розроблений програмний продукт являє собою веб-застосування. Для коректної роботи системи інформаційної підтримки навчальних курсів до технічних засобів висуваються наступні вимоги:

1) Комп’ютер або пристрій, на якому буде використовуватись дана система, повинна володіти наступними характеристиками:
   - Тактова частота процесора не нижче 1ГГц;
   - Об’єм оперативної пам’яті не менше 4 Гб;
   - Підключення до інтернету.
2) Необхідний один із перелічених нежче веб-браузерів:
   - Google Chrome;
   - Microsoft Edge;
   - Mozilla Firefox;
   - Opera 11 і вище

4.3 Архітектура програмного забезпечення

4.3.1 Діаграма класів

Під час розробки даного програмного продукту всього було створено близько 65 класів. Усі класи розподілені по пакетах, відповідно до їх призначення. В ході розробки було створено 10 пакетів. На рисунку 4.1 зображено діаграму пакетів розробленого застосунку.

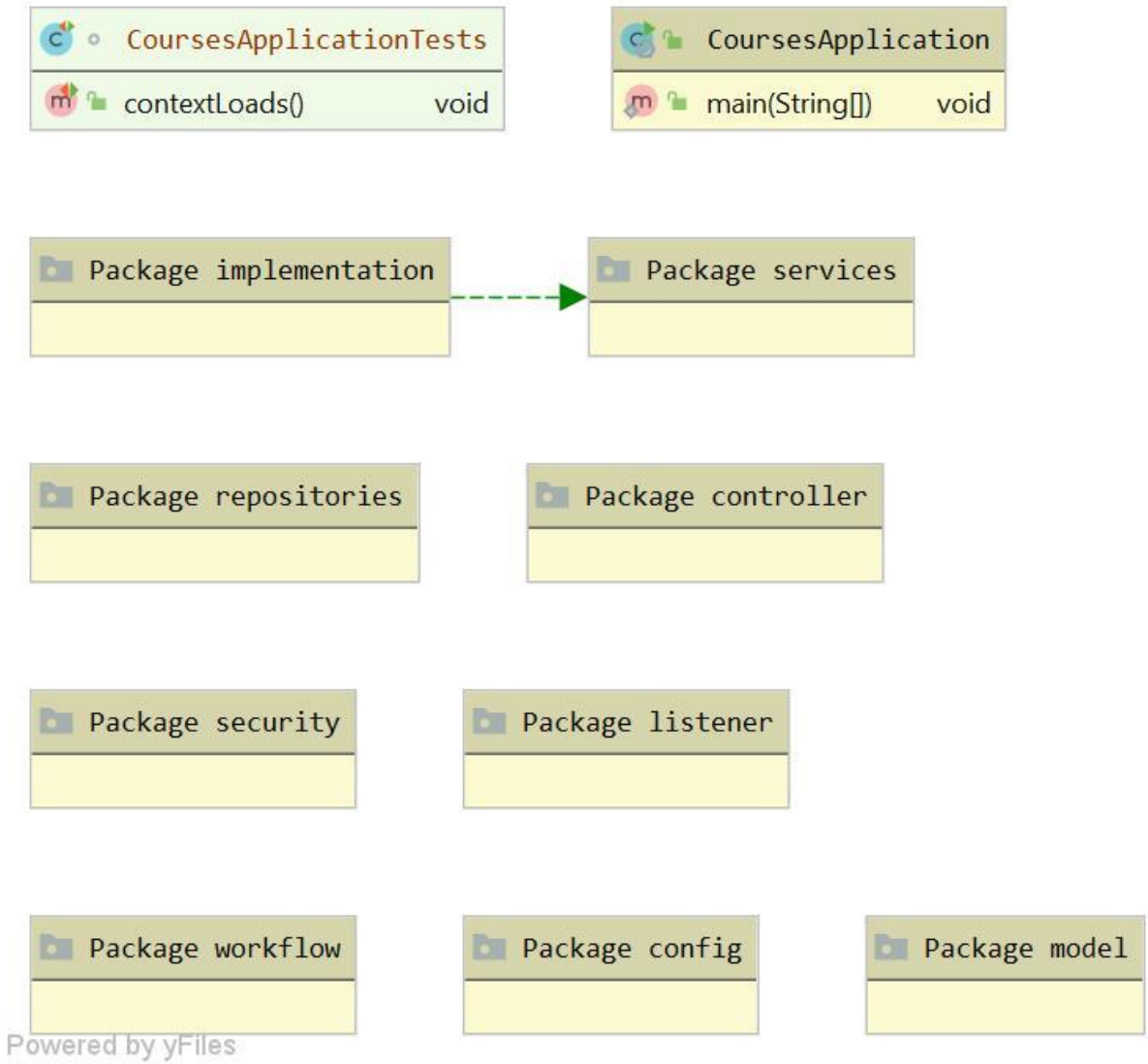


Рисунок 4.1 – Діаграма пакетів

Нижче наведено опис кожного з пакетів:

1) config – даний пакет містить класи, в яких описані налаштування програмного продукту, а саме: визначення рівнів доступу для користувачів з різними ролями; шифрування паролів користувачів; налаштування з'єднання клієнтської та серверної частини через Websocket.

2) controller – даний пакет містить класи-контролери, які обробляють запити користувачів та повертають їм відповідь.

3) Services – містить інтерфейси, в яких визначено методи, що мають бути реалізовані в сервісах.

4) implementation – містить класи-сервіси, які реалізовують інтерфейси з пакету services.

5) model – містить класи-сутності, які використовуються в даній розробці. Даний пакет містить сутності двох типів: DTO (Data Transfer Object - об'єкти, які пересилаються між клієнтською та серверною частиною) та entity (сутності, які зберігаються в базі даних).

6) listener – містить класи, в яких визначено дії, які мають автоматично відбуватись при запуску програми.

7) security – містить класи, в яких реалізована логіка, що визначає статус та роль авторизованих користувачів.

8) repositories – містить класи, які відповідають за з'єднання з базою даних.

9) workflow – містить класи, які запускають програму та визначають дані за замовчуванням, які мають бути завантажені в базу даних при запуску програми.

10) resourses – містить шаблони веб-сторінок, які користувач буде бачити у своєму браузері.

Схема структурна класів програмного забезпечення зображена у графічному матеріалі.

4.3.2 Діаграма послідовності

Схема структурна послідовності для процесу реєстрації в системі, авторизації та реєстрації на курс представлена в графічному матеріалі. На ній показано взаємодію веб-сторінок, класів-контролерів, класів-сервісів та бази даних.

Користувач натискає відповідну кнопку на веб-сторінці і відправляє запит на сервер. Запит приймає клас-контролер та звертається до відповідного сервісу. Сервіс приймає дані від контролера та зберігає їх у базі даних (або отримує з неї дані, залежно від типу запиту), а контролер надсилає відповідь користувачу.

4.3.3 Діаграма компонентів

Схема структурна компонентів зображена у графічному матеріалі. Вона складається із трьох компонентів:

1) База даних, в якій зберігаються усі дані системи.
2) Бекенд-сервер, який опрацьовує запити користувачів та взаємодіє з базою даних. Він містить у собі 5 основних пакетів: controller, model, repositories, services, resourses. Дані пакети були детально описані в підпункті 4.3.2.
3) Веб-браузер, який відображає сторінки програмного продукту.

4.3.4 Специфікація функцій

Основні функції класів даного програмного забезпечення описані в таблицях 4.1 – 4.11.

Таблиця 4.1 – Функції класу UserServiceImpl

Назва	Опис
findAll()	Повертає всіх користувачів системи.
createUser(UserDto userDto)	Створює нового користувача.
deleteUser(UUID userId)	Видаляє користувача по його id.
updateUser(UserDto userDto)	Оновлює інформацію про користувача.
findById(UUID userId)	Повертає користувача по його id
findByEmail(String email)	Повертає користувача по його електронній адресі.
findAllByGroup(UUID groupId)	Повертає всіх користувачів, які входять до групи, яка має ідентифікатор groupId.
findManagerBySubordinateId(UUID subordinateId)	Повертає менеджера, за яким закріплений студент з ідентифікатором subordinateId.
findFreeUsers()	Повертає всіх користувачів, у яких немає менеджера.
setManagerId(UUID managerId, UUID userIds)	Закріплює менеджера за студентом.
getRoles()	Повертає всі ролі, які є в системі.
assignUsersToGroup(UUID groupId, UUID[] userIds)	Реєструє користувачів як учасників групи.

Таблиця 4.2 – Функції класу CourseServiceImpl

Назва	Опис
createCourse(CourseDto courseDto)	Створює новий курс
updateCourse(UUID courseId, CourseDto courseDto)	Оновлює інфомрацію про курс
deleteById(UUID id)	Видаляє курс із заданим id
findById(UUID courseId)	Повертає курс по його id
findAll()	Повертає всі курси
findByTeacherId(UUID id)	Повертає курс, який веде викладач із заданим id
findCoursesByUser(UserDto userDto)	Повертає всі курси, на які зареєстрований користувач
findCourseIdByGroupId(UUID groupId)	Повертає курс, до якого належить задана група

Таблиця 4.3 – Функції класу GroupServiceImpl

Назва	Опис
createGroup(GroupDto groupDto)	Створює нову групу
updateGroup(UUID groupId, GroupDto groupDto)	Оновлює інформацію про групу
findById(UUID groupId)	Повертає групу по id
deleteById(UUID groupId)	Видаляє групу по id
findAll()	Повертає список всіх груп
findAllByCourseId(String courseId)	Повертає список всіх груп, які відносяться до заданого курсу

Продовження таблиці 4.3

findAllGroupsWithoutChatByCourseId(String courseId)	Повертає всі групи, у яких немає групового чату
deleteFromUserGroupByUserIdAndGroupId(String userId, UUID groupId)	Видаляє студента із групи

Таблиця 4.4 – Функції класу LessonServiceImpl

Назва	**Опис**
createLesson(LessonDto lessonDto)	Створює нове заняття
updateLesson(UUID lessonId, LessonDto lessonDto)	Оновлює інформацію про заняття
deleteById(UUID lessonId)	Видаляє заняття по id
findAll()	Повертає список усіх занять
findById(UUID lessonId)	Повертає заняття із заданим id
getLessonsByCourseId(UUID courseId)	Повертає всі заняття для заданого курсу

Таблиця 4.5 – Функції класу ScheduleServiceImpl

Назва	**Опис**
findById(UUID scheduleId)	Повертає розклад занять по id
findAll()	Повертає розклад всіх занять
deleteById(UUID scheduleId)	Видаляє розклад занять
updateSchedule(UUID scheduleId, ScheduleDto scheduleDto)	Оновлює розклад занять
createSchedule(ScheduleDto scheduleDto)	Створює розклад занять

Таблиця 4.6 – Функції класу ChatServiceImpl

Назва	Опис
findById(String id)	Повертає чат по заданому id
createChat(Chat entity)	Створює новий чат
update(Chat entity)	Оновлює інфомрацію про чат
deleteById(String id)	Видаляє чат по id
findAll()	Повертає всі чати
getChatNameById(UUID chatId)	Повертає назву чату по його id
createGroupChat(Chat chat)	Створює груповий чат
getChatByGroupId(String groupId)	Повертає груповий чат по id
ifChatExistsByUsersId(String senderId, String receiverId)	Повертає true, у випадку якщо існує чат між двома користувачами, і false – якщо такий чат відсутній
findChatByName(String chatName)	Повертає чат по його назві
findChatByUsersId(String senderId, String receiverId)	Повертає чат між двома заданими користувачами
findAllPersonalChatsByUserId(String userId)	Повертає всі особисті чати для заданого користувача

Таблиця 4.7 – Функції класу ChatMessageService

Назва	Опис
saveMessage(ChatMessage chatMessage)	Зберігає повідомлення
getMessagesByChatId(String chatId)	Повертає всі повідомлення для заданого чату

Таблиця 4.8 – Функції класу ProfileServiceImpl

Назва	Опис
initManager(UserDto user)	Повертає менеджера, який закріплений за студентом
getReadableRole(UserDto user)	Повертає роль користувача, яка буде відображатись на його сторінці

Таблиця 4.9 – Функції класу UserGroupServiceImpl

Назва	Опис
getUserGroupListByGroupId(String groupId)	Повертає список студентів, які є учанисками заданої групи
setUsersToGroup(UUID groupId, UUID[] userId)	Записує заданих студентів до групи
deleteFromUserGroupByUserIdAndGroupId(String userId, String groupId)	Видаляє студента із заданої групи

Таблиця 4.10 – Функції класу OrderServiceImpl

Назва	Опис
createOrder(Order order)	Створює нову заяву студента на реєстрацію на курс
findById(String orderId)	Повертає заяву на реєстрацію по id
deleteById(String orderId)	Видаляє заяву по id
findAll()	Повертає всі заяви на реєстрацію на курси
findAllByCourseId(String courseId)	Повертає всі заяви на реєстрацію на заданий курс
findAllByUserId(String userId)	Повертає всі заяви на реєстрацію

Таблиця 4.11 – Функції класу AttendanceServiceImpl

Назва	Опис
createAttendance(AttendanceDto attendanceDto)	Створює об’єкт, який містить інформацію про відвідуваність занять студентом
updateAttendance(UUID attendanceId, AttendanceDto attendanceDto)	Оновлює відвідуваність занять студентом
findById(UUID attendanceId)	Повертає відвідуваність заняття по id
deleteById(UUID attendanceId)	Видаляє відвідуваність по id
findAll()	Повертає усі відвідуваності занять
getAttendanceListWithStudents(ScheduleDto schedule, List<UserDto> users)	Повертає відвідуваність занять для заданих студентів

4.3.1 Маршрутизація

Сервер приймає запити від клієнтської частини, опрацьовує їх і надсилає відповідь користувачу. За взаємодію клієнтської та серверної частини відповідає пакет *controller*, який містить в собі класи-контролери Нижче описано, як класи відповідають на запити до різних URL адрес

Маршрути REST API наведені в таблицях 4.12 – 4.22.

Таблиця 4.12 – Маршрути класу MainController

Маршрут	Метод	Дії сервера
/, /index	GET	Повертає головну сторінку застосунку
/admin	GET	Повертає сторінку адміністратора
/login	GET	Повертає сторінку авторизації
/profilepage	GET	Повертає сторінку профіля користувача

Таблиця 4.13 – Маршрути класу UserController

Маршрут	Метод	Дії сервера
/registration	GET	Повертає сторінку реєстрації

Таблиця 4.14 – Маршрути класу TeacherController

Маршрут	Метод	Дії сервера
/trainers/{page}	GET	Повертає сторінку з усіма викладачами, які є в системі
/trainer-profile/{id}	GET	Повертає сторінку викладача
/trainer-add	GET	Повертає сторінку створення нового облікового запису для викладача
/trainer-update/{id}	GET	Повертає сторінку для редагування інформації про викладача
/trainer-update/{id}	POST	Зберігає оновлену інформацію про викладача в базі даних
/saveTrainer	POST	Зберігає новий обліковий запис викладача в базі даних

Таблиця 4.15 – Маршрут класу ProfileController

Маршрут	Метод	Дії сервера
/profile_page/{userId}	GET	Повертає сторінку профіля користувача

Таблиця 4.16 – Маршрути класу ManagerController

Маршрут	Метод	Дії сервера
/managers/{page}	GET	Повертає сторінку з усіма менеджерами, які є в системі
/manager-profile/{id}	GET	Повертає сторінку профіля менеджера
/subordinates/{page}/{userId}	GET	Повертає сторінку зі студентами, за якими закріплений заданий менеджер
/manager-update/{id}	GET	Повертає сторінку для оновлення інформації про менеджера
/manager-add	GET	Повертає сторінку створення нового облікового запису для менеджера
/manager-update/{id}	POST	Зберігає оновлену інформацію про менеджера в базі даних
/saveManager	POST	Зберігає новий обліковий запис менеджера в базі даних

Таблиця 4.17 – Маршрути класу ManagerController

Маршрут	Метод	Дії сервера
/managers/{page}	GET	Повертає сторінку з усіма менеджерами, які є в системі
/manager-profile/{id}	GET	Повертає сторінку профіля менеджера
/subordinates/{page}/{userId}	GET	Повертає сторінку зі студентами, за якими закріплений заданий менеджер

Продовження таблиці 4.17

/manager-update/{id}	GET	Повертає сторінку для оновлення інформації про менеджера
/manager-add	GET	Повертає сторінку створення нового облікового запису для менеджера
/manager-update/{id}	POST	Зберігає оновлену інформацію про менеджера в базі даних
/saveManager	POST	Зберігає новий обліковий запис менеджера в базі даних

Таблиця 4.18 – Маршрути класу LessonController

Маршрут	**Метод**	**Дії сервера**
/course_lessons/{page}/{courseId}	GET	Повертає сторінку з усіма заняттями для заданого курсу
/lesson-add/{courseId}	GET	Повертає сторінку створення новго заняття
/lesson-save	POST	Зберігає нове заняття в базі даних
/edit-lesson/{lessonId}/{id}	GET	Повертає сторінку редагування інформації про заняття
/lesson-delete-by/{lessonId}/{id}	GET	Видаляє задане заняття

Таблиця 4.19 – Маршрути класу GroupController

Маршрут	**Метод**	**Дії сервера**
/group_create/{page}/{courseId}	GET	Повертає сторінку з усіма групами для певного курсу

Продовження таблиці 4.19

/create_group_chats/{page}/{courseId}	GET	Повертає сторінку створення групового чату
/group_users/{page}/{groupId}	GET	Повертає сторінку з усіма учасниками заданої групи
/add_users_to_group/{groupId}	GET	Повертає сторінку для додавання студентів до групи
/add_selected_users_to_group/{groupId}	POST	Додає студентів до заданої групи
/group-add/{courseId}	GET	Повертає сторінку створення нової групи
/group-save	POST	Зберігає нову групу в базі даних
/edit-group/{groupId}/{id}	GET	Повертає сторінку редагування групи
/group-delete-by/{groupId}/{id}	POST	Видаляє задану групу

Таблиця 4.20 – Маршрути класу CourseController

Маршрут	**Метод**	**Дії сервера**
/coursepage	GET	Повертає сторінку з усіма курсами
/course-add	GET	Повертає сторінку створення нового курсу
/course-save	POST	Зберігає новий курс в базі даних
/edit-course-{id}	GET	Повертає сторінку редагування курсу
/course-info/{courseId}	GET	Повертає сторінку з курсом
/join-course/{courseId}/{userId}	POST	Зберігає заявку студента на реєстрацію в базі даних

Продовження таблиці 4.20

/edit-course/{id}	POST	Зберігає оновлений курс в базі даних
/course-delete-by/{id}	POST	Видаляє заданий курс

Таблиця 4.21 – Маршрути класу LessonController

Маршрут	**Метод**	**Дії сервера**
/chat.sendMessage	GET	Відправляє повідомлення в чат та зберігає його в базі даних
/chat.addUser	GET	Додає користувача в чат
/chat/{chatId}	GET	Повертає сторінку з особистим чатом
/show_chats/{page}	GET	Повертає сторінку з усіма чатами
/create-chat	GET	Повертає сторінку створення нового чату
/create-chat-for-groups/{courseId}	POST	Зберігає груповий чат в базі даних
/save-chat	POST	Зберігає особистий чат в базі даних
/open-group-chat/{groupId}	GET	Повертає сторінку з груповим чатом
/open-personal-chat-list	GET	Повертає сторінку з усіма особистими чатами

Таблиця 4.22 – Маршрути класу AttendanceController

Маршрут	**Метод**	**Дії сервера**
/myCourses	GET	Повертає сторінку з усіма курсами, які веде викладач
/myGroups/{courseId}	GET	Повертає сторінку з групами, у яких веде заняття викладач

Продовження таблиці 4.22

/myScheduleForGroup/{groupId}	GET	Повертає сторінку з розкладом для груп, у яких веде заняття викладач
/presence/{groupId}/{scheduleId}	GET	Повертає сторінку для перевірки відвідуваності занять студентами
/submit-attendance	POST	Зберігає інформацію про відвідуваність занять в базі даних

4.4 Опис звітів

Основна інформація, яка стосується навчального процесу, доступна в особистому кабінеті користувача. Зі свого профілю користувач може переглянути список курсів, на які він зареєстрований, відвідати сторінку свого менеджера або викладача, переглянути свої повідомлення та список рекомендованих курсів. Сторінка профілю користувача зображена на рисунку 4.1.

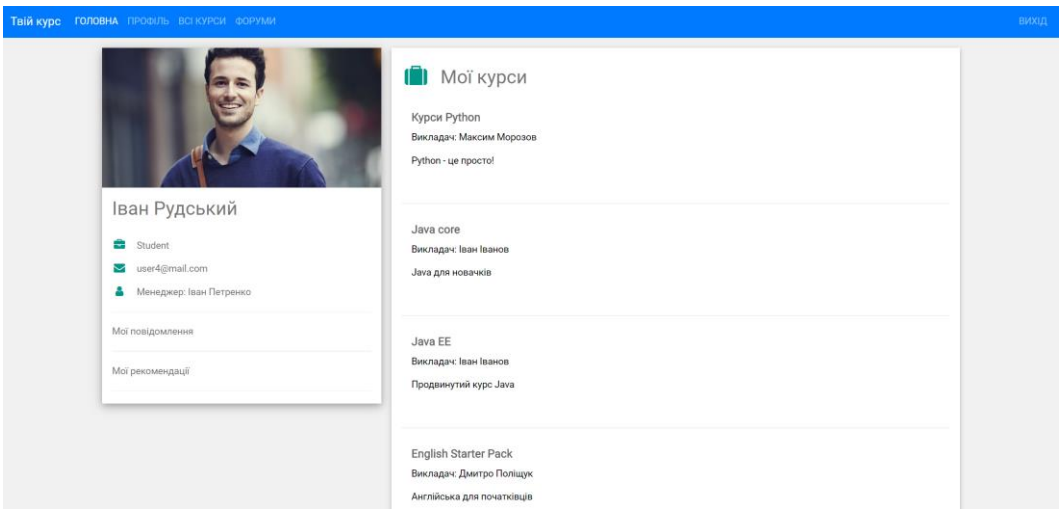


Рисунок 4.1 – Профіль користувача

На рисунку 4.2 зображено сторінку з курсами, які система рекомендує на основі інформації про інші курси, на які зареєстрований користувач.

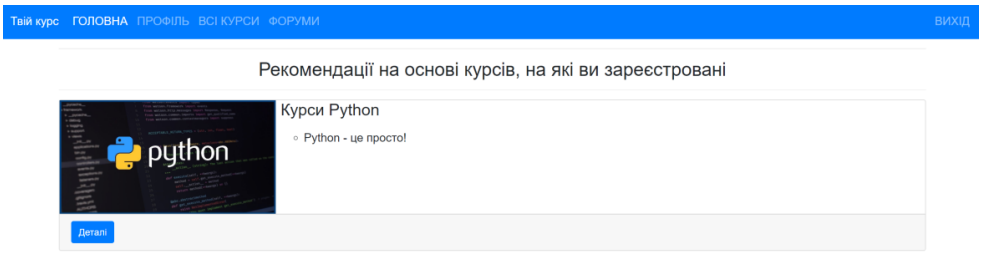


Рисунок 4.2 – Сторінка з рекомендованими курсами

ВИСНОВОК ДО РОЗДІЛУ

В даному розділі були детально описані засоби розробки, які використовувались під час створення програмного продукту, вказана їхня характеристика та особливості. Вказані вимоги до технічного забезпечення та загальні вимоги. Була описана архітектура розробленого застосунку у вигляді діаграми пакетів, фрагменту діаграми класів, діаграми компонентів та послідовності. Також в даному розділі була надана детальна специфікація функцій, описано маршрути REST API, які використовуються в даному проєкті та наведено приклад звіту.

5 ТЕХНОЛОГІЧНИЙ РОЗДІЛ

5.1 Керівництво користувача

Для запуску даного програмного продукту необхідно ввести в пошуковій стрічці браузера наступну адресу: «localhost:8080».

Система передбачає 5 типів користувачів, а саме: гість, студент, адміністратор, менеджер, викладач. На рисунку 5.1 зображена головна сторінка застосунку, на яку потрапляє незареєстрований користувач. На ній для користувача доступний такий функціонал: вхід у систему та реєстрація.

Рисунок 5.1 – Головна сторінка застосунку

Для того, щоб отримати доступ до функцій системи, користувачу необхідно зареєструватися та авторизуватися. Для реєстрації потрібно вказати ім’я, прізвище, електронну адресу та пароль. Форма реєстрації зображена на рисунку 5.2.

Рисунок 5.2 - Форма для реєстрації

Після успішної реєстрації користувач може авторизуватися в системі. Для цього йому необхідно ввести свою електронну адресу та пароль, які він вказував при реєстрації. Форма авторизації зображена на рисунку 5.3

Рисунок 5.3 – Форма авторизації

У разі введення користувачем невірних даних під час авторизації, система надає повідомлення про помилку, як зображено на рисунку 5.4.

Рисунок 5.4 – Повідомлення про помилку під час авторизації

Після успішної авторизації користувачу стають доступні усі функції системи. З будь-якої сторінки він може відкрити свій профіль, список курсів та список чатів, а також вийти із системи. На рисунку 5.5 зображена сторінка, яка містить інформацію про всі курси.

Рисунок 5.5 – Сторінка з курсами

Користувач може відкрити сторінку курсу, переглянути детальну інформацію та зареєструватись на нього. Після цього заявка буде збережена в базі даних та опрацьована адміністратором.

Рисунок 5.6 – Інформація про курс

В профілі користувача відображається основна інформація про нього: ім’я, електронна адреса, менеджер, список курсів, на які він зареєстрований. Зі свого профілю користувач може перейти до рекомендацій, особистих повідомлень, відкрити профіль свого менеджера або викладача, який веде курс. На рисунку 5.7 зображена сторінка профілю користувача.

Рисунок 5.7 – Профіль користувача

При натисканні вкладки «Мої рекомендації» користувач перейде на сторінку із курсами, які система рекомендує на основі даних про інші курси, на які він зареєстрований. Сторінка із рекомендаціями зображена на рисунку 5.8

Рисунок 5.8 – Сторінка з рекомендованими курсами

Адміністратор системи має можливість створювати, редагувати та видаляти курси і групи. Також він може переглядати список занять та розклад, додавати нових студентів до груп. При натисканні адміністратором кнопки «Курси», він потрапить на сторінку з курсами, яка зображена на рисунку 5.9.

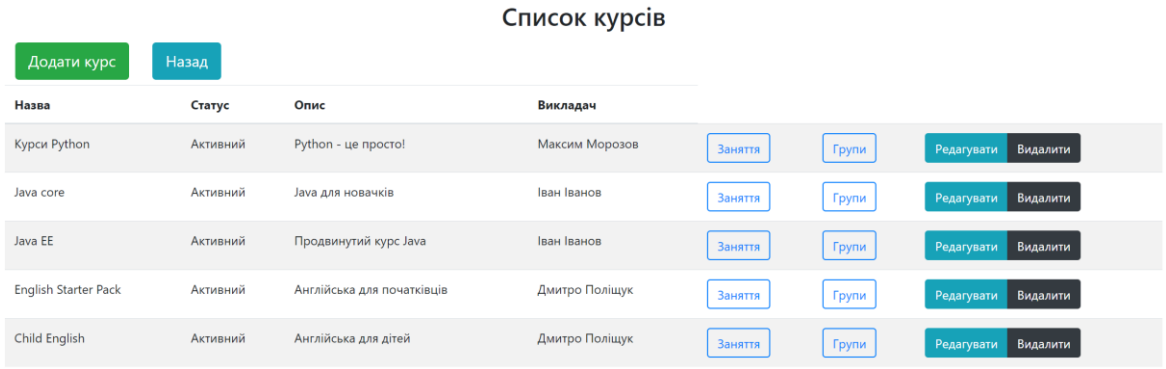


Рисунок 5.9 – Сторінка куревання курсами

Після натискання кнопки «Групи», для адміністратора відкриється сторінка, що містить інформацію про всі групи, які належать до даного курсу. Адміністратор може їх редагувати та видаляти, переглянути студентів у групах та розклад занять. Сторінка з групами щображена на рисунку 5.10.

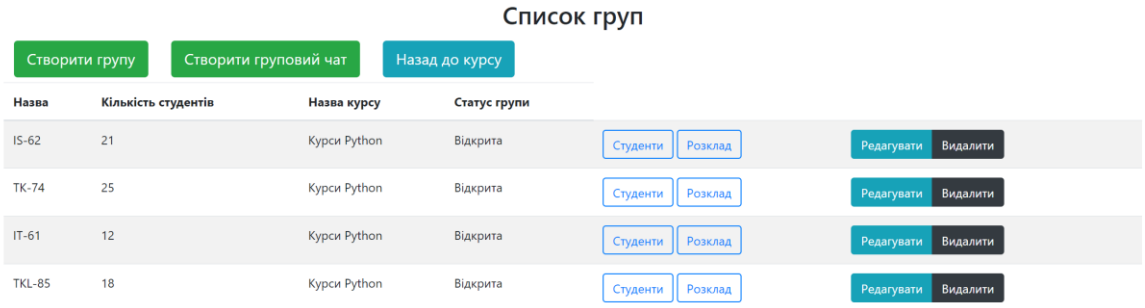


Рисунок 5.10 – Сторінка керування групами

5.2 Випробування програмного продукту

В цьому підрозділі наведено опис тестів і порядок їх виконання для перевірки відповідності програмного забезпечення функціональним вимогам, представленим у технічному завданні на створення системи інформаційної підтримки навчальних курсів.

5.2.1 Мета випробувань

Метою випробувань являється перевірка відповідності функцій системи інформаційної підтримки навчальних курсів вимогам технічного завдання.

5.2.2 Загальні положення

Випробування проводяться на основі наступних документів:

– ГОСТ 34.603–92. Інформаційна технологія. Види випробувань автоматизованих систем;
– ГОСТ РД 50-34.698-90. Автоматизовані системи вимог до змісту документів.

5.2.3 Результати випробувань

Під час проведення тестування була перевірена функціональність системи. А таблицях 5.1 – 5.17 наведено перелік випробувань функціональних можливостей системи.

Таблиця 5.1 – Авторизація

Назва	Перевірка функції «Авторизація»
Початковий стан системи	Відкрита сторінка «Вхід»
Вхідні дані	Електронна адреса та пароль користувача

Продовження таблиці 5.1

Схема проведення тесту	Ввести у поле «Пошта» набір символів, структура якого не відповідає структурі електронної адреси; у поле «Пароль» - пароль, довжина якого не менша 6 символів. Натиснути кнопку «Вхід»
Очікуваний результат	Повідомлення про те, що введено некоректну електронну адресу. Вхід у систему не виконано
Стан системи після проведення випробувань	Відкрита сторінка «Вхід» та виведено повідомлення про помилку

Таблиця 5.2 – Авторизація

Назва	**Перевірка функції «Авторизація»**
Початковий стан системи	Відкрита сторінка «Вхід»
Вхідні дані	Електронна адреса та пароль користувача
Схема проведення тесту	Ввести у поле «Пошта» випадкову електронну адресу, яка має коректну структуру; у поле «Пароль» - коректний пароль
Очікуваний результат	Повідомлення про те, що введено невірну електронну адресу або пароль. Вхід у систему не виконано
Стан системи після проведення випробувань	Відкрита сторінка «Вхід» та виведено повідомлення про помилку

Таблиця 5.3 – Авторизація

Назва	**Перевірка функції «Авторизація»**
Початковий стан системи	Відкрита сторінка «Вхід»
Вхідні дані	Електронна адреса та пароль користувача
Схема проведення тесту	Ввести у поле «Пошта» правильну електронну адресу, яка має коректну структуру; у поле «Пароль» - випадковий набір символів
Очікуваний результат	Повідомлення про те, що введено невірну електронну адресу або пароль. Вхід у систему не виконано
Стан системи після проведення випробувань	Відкрита сторінка «Вхід» та виведено повідомлення про помилку

Таблиця 5.4 – Авторизація

Назва	**Перевірка функції «Авторизація»**
Початковий стан системи	Відкрита сторінка «Вхід»
Вхідні дані	Електронна адреса та пароль користувача
Схема проведення тесту	Ввести у поле «Пошта» правильну електронну адресу, яка має коректну структуру; у поле «Пароль» - правильний пароль, який відповідає даному обліковому запису
Очікуваний результат	Виконано вхід у систему
Стан системи після проведення випробувань	Вхід у систему виконано, відкрита головна сторінка

Таблиця 5.5 – Реєстрація

Назва	**Перевірка функції «Реєстрація»**
Початковий стан системи	Відкрита сторінка «Реєстрація»
Вхідні дані	Ім’я, прізвище, електронна адреса та пароль користувача
Схема проведення тесту	Ввести у поле «Ім’я» та «Прізвище» коректні значення, які відповідають реальним даним користувача; у поле «Пошта» - електронну адресу, яка вже зареєстрована в системі; у поле «Пароль» - коректний пароль, довжина якого більша шести символів
Очікуваний результат	Повідомлення про те, що така електронна адреса вже зареєстрована в системі. Реєстрація не виконана
Стан системи після проведення випробувань	Відкрита сторінка «Реєстрація» та виведено повідомлення про помилку

Таблиця 5.6 – Реєстрація

Назва	**Перевірка функції «Реєстрація»**
Початковий стан системи	Відкрита сторінка «Реєстрація»
Вхідні дані	Ім’я, прізвище, електронна адреса та пароль користувача

Продовження таблиці 5.6

Схема проведення тесту	У поле «Пошта» ввести унікальну коректну електронну адресу; поля «Ім’я» та «Прізвище» залишити пустими; у поле «Пароль» - коректний пароль
Очікуваний результат	Повідомлення про те, що всі поля повинні бути заповнені. Реєстрація не виконана
Стан системи після проведення випробувань	Відкрита сторінка «Реєстрація» та виведено повідомлення про помилку

Таблиця 5.7 – Реєстрація

Назва	**Перевірка функції «Реєстрація»**
Початковий стан системи	Відкрита сторінка «Реєстрація»
Вхідні дані	Ім’я, прізвище, електронна адреса та пароль користувача
Схема проведення тесту	У поле «Пошта» ввести унікальну коректну електронну адресу; в поля «Ім’я» та «Прізвище» ввести коректні значення, які відповідають реальним даним користувача; у поле «Пароль» ввести набір символів, довжина якого менша шести
Очікуваний результат	Повідомлення про те, що пароль повинен складатися щонайменш із 6 символів. Реєстрація не виконана
Стан системи після проведення випробувань	Відкрита сторінка «Реєстрація» та виведено повідомлення про помилку

Таблиця 5.8 – Реєстрація

Назва	**Перевірка функції «Реєстрація»**
Початковий стан системи	Відкрита сторінка «Реєстрація»
Вхідні дані	Ім’я, прізвище, електронна адреса та пароль користувача
Схема проведення тесту	У поле «Пошта» ввести випадковий набір символів, структура якого не відповідає структурі електронної адреси; в поля «Ім’я» та «Прізвище» ввести коректні значення, які відповідають реальним даним користувача; у поле «Пароль» ввести коректний пароль, довжина якого більша шести символів
Очікуваний результат	Повідомлення про те, що введено некоректну електронну адресу. Реєстрація не виконана
Стан системи після проведення випробувань	Відкрита сторінка «Реєстрація» та виведено повідомлення про помилку

Таблиця 5.9 – Реєстрація

Назва	**Перевірка функції «Реєстрація»**
Початковий стан системи	Відкрита сторінка «Реєстрація»
Вхідні дані	Ім’я, прізвище, електронна адреса та пароль користувача
Схема проведення тесту	Всі поля залишити пустими
Очікуваний результат	Повідомлення про те, що користувач не ввів жодних даних. Реєстрація не виконана

Продовження таблиці 5.9

Стан системи після проведення випробувань	Відкрита сторінка «Реєстрація» та виведено повідомлення про помилку

Таблиця 5.10 – Створення курсу

Назва	**Перевірка функції «Реєстрація»**
Початковий стан системи	Відкрита сторінка «Новий курс»
Вхідні дані	Назва курсу, статус, опис, викладач
Схема проведення тесту	Всі поля залишити пустими
Очікуваний результат	Повідомлення про те, що користувач не ввів жодних даних. Курс не створений
Стан системи після проведення випробувань	Відкрита сторінка «Новий курс» та виведено повідомлення про помилку

Таблиця 5.11 – Створення курсу

Назва	**Перевірка функції «Реєстрація»**
Початковий стан системи	Відкрита сторінка «Новий курс»
Вхідні дані	Назва курсу, статус, опис, викладач
Схема проведення тесту	Поле «Назва курсу» залишити пустим; поля «Статус», «Опис», «Викладач» заповнити коректними даними
Очікуваний результат	Повідомлення про те, що користувач не вказав назву курсу. Курс не створений
Стан системи після проведення випробувань	Відкрита сторінка «Новий курс» та виведено повідомлення про помилку

Таблиця 5.12 – Створення курсу

Назва	**Перевірка функції «Реєстрація»**
Початковий стан системи	Відкрита сторінка «Новий курс»
Вхідні дані	Назва курсу, статус, опис, викладач
Схема проведення тесту	Поля «Назва курсу», «Статус», «Опис», заповнити коректними даними; поле «Викладач» залишити пустим
Очікуваний результат	Повідомлення про те, що користувач не вказав викладача для курсу. Курс не створений
Стан системи після проведення випробувань	Відкрита сторінка «Новий курс» та виведено повідомлення про помилку

Таблиця 5.13 – Створення групи

Назва	**Перевірка функції «Реєстрація»**
Початковий стан системи	Відкрита сторінка «Нова група»
Вхідні дані	Назва групи, кількість студентів, статус
Схема проведення тесту	Всі поля залишити пустими
Очікуваний результат	Повідомлення про те, що користувач не ввів жодних даних. Група не створена
Стан системи після проведення випробувань	Відкрита сторінка «Нова група» та виведено повідомлення про помилку

Таблиця 5.14 – Створення групи

Назва	**Перевірка функції «Реєстрація»**
Початковий стан системи	Відкрита сторінка «Нова група»
Вхідні дані	Назва групи, кількість студентів, статус
Схема проведення тесту	В поле «Назва групи» ввести коректну назву; в полі «Кількість студентів» ввести нуль
Очікуваний результат	Повідомлення про те, що не можна створити групу, кількість студентів в якій рівна нулю. Група не створена
Стан системи після проведення випробувань	Відкрита сторінка «Нова група» та виведено повідомлення про помилку

Таблиця 5.15 – Створення заняття

Назва	**Перевірка функції «Реєстрація»**
Початковий стан системи	Відкрита сторінка «Нове заняття»
Вхідні дані	Назва заняття, опис, тривалість
Схема проведення тесту	Всі поля залишити порожніми
Очікуваний результат	Повідомлення про те, що користувач не вказав жодних даних. Заняття не створено
Стан системи після проведення випробувань	Відкрита сторінка «Нове заняття» та виведено повідомлення про помилку

Таблиця 5.16 – Створення заняття

Назва	**Перевірка функції «Реєстрація»**
Початковий стан системи	Відкрита сторінка «Нове заняття»
Вхідні дані	Назва заняття, опис, тривалість
Схема проведення тесту	В полі «Назва заняття» вказати коректну назву; в полі «Опис» вказати коректний опис заняття; в поле «Тривалість» вказати нуль
Очікуваний результат	Повідомлення про те, що тривалість заняття не може бути нульовою. Заняття не створено
Стан системи після проведення випробувань	Відкрита сторінка «Нове заняття» та виведено повідомлення про помилку

Таблиця 5.17 – Створення заняття

Назва	**Перевірка функції «Реєстрація»**
Початковий стан системи	Відкрита сторінка «Нове заняття»
Вхідні дані	Назва заняття, опис, тривалість
Схема проведення тесту	В полі «Назва заняття» вказати коректну назву; в полі «Опис» вказати коректний опис заняття; в полі «Тривалість» вказати невід’ємне число
Очікуваний результат	Створено нове заняття. Відкрита сторінка усіх занять
Стан системи після проведення випробувань	Створено нове заняття. Відкрита сторінка усіх занять

ВИСНОВОК ДО РОЗДІЛУ

В даному розділі було наведено керівництво користувача для використання розробленого застосунку, описано основні функції, які були реалізовані відповідно до постановки задачі, та представлено зображення екранних форм. Були розроблені тести для перевірки коректності роботи програми та відповідності програмного забезпечення функціональним вимогам, представленим у технічному завданні. Для кожного тесту наведено вхідні дані, очікувані та фактичні стани системи, описано результати випробувань.

ЗАГАЛЬНІ ВИСНОВКИ

В ході виконання даного дипломного проєкту було проаналізовано предметну область, проблематику та актуальність поставленої задачі. Також була здійснена порівняльна характеристика існуючих аналогів. Після вивчення даних матеріалів було визначено основні вимоги до програмного продукту, функції, які він повинен виконувати, та його призначення в цілому. Результатом роботи є розроблена система інформаційної підтримки навчальних курсів.

Предметне середовище, опис процесу діяльності, існуючі аналоги, мета та цілі розробки були детально описані в розділі «Загальні положення».

В розділі «Інформаційне забезпечення» було описано вхідні та вихідні дані, з якими працює система. Також продемонстрована структура бази даних та надана характеристика усіх сутностей.

Для розробки системи рекомендацій навчальних курсів було використано алгоритм колаборативної фільтрації на основі подібності елементів під назвою «Item-to-Item». Даний алгоритм детально описаний в розділі «Математичне забезпечення».

Засоби розробки, які використовувались під час створення програмного продукту, описані в розділі «Програмне та технічне забезпечення». Основними засобами розробки є мова програмування Java, фреймворк Spring Boot та нереляційна база даних MongoDB.

Також в даній роботі було надано керівництво користувача, в якому описано основні функції системи та інструкції щодо використання. Наведено зображення екранних форм та надано їх короткий опис.

Створений веб-застосунок дає можливість усім учасникам навчального процесу зберігати необхідну інформацію та обмінюватись нею в межах одного середовища. Це робить навчання простішим та ефективнішим.

ПЕРЕЛІК ПОСИЛАНЬ

1. Coursera [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Coursera
2. Moodle [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Moodle
3. Moodle [Електронний ресурс] – Режим доступу: https://habr.com/ru/company/teachbase/blog/369321/
4. MongoDB [Електронний ресурс] – Режим доступу: https://www.lessons-tva.info/articles/informat/42.html
5. Рекомендаційна система [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BA%D0%BE%D0%BC%D0%B5%D0%BD%D0%B4%D0%B0%D1%86%D1%96%D0%B9%D0%BD%D0%B0_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0
6. Рекомендательные системы. Задачи коллаборативной фильтрации [Електронний ресурс] – Режим доступу: http://www.machinelearning.ru/wiki/index.php?title=%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D0%B5_%28%D0%BA%D1%83%D1%80%D1%81_%D0%BB%D0%B5%D0%BA%D1%86%D0%B8%D0%B9%2C_%D0%9A.%D0%92.%D0%92%D0%BE%D1%80%D0%BE%D0%BD%D1%86%D0%BE%D0%B2%29
7. Колаборативна фільтрація [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BB%D0%B0%D0%B1%D0%BE%D1%80%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D0%B0_%D1%84%D1%96%D0%BB%D1%8C%D1%82%D1%80%D0%B0%D1%86%D1%96%D1%8F
8. Алгоритм колаборативної фільтрації Item-to-Item [Електронний ресурс] – Режим доступу: https://ru.wikipedia.org/wiki/Slope_One
9. Косинусна подібність [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D1%81%D0%B8%D0%BD%D1%83%D1%81_%D0%BF%D0%BE%D0%B4%D1%96%D0%B1%D0%BD%D0%BE%D1%81%D1%82%D1%96

10. Мусієнко В.С., Ковтунець О.В. Метод колаборативної фільтрації на основі подібності елементів / Всеукраїнська науково-практична конференція молодих вчених та студентів «Інформаційні системи та технології управління» (ІСТУ-2020) – м. Київ.: НТУУ «КПІ ім. Ігоря Сікорського», 24 та 30 квітня. – С. 101-105.
11. Java [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Java
12. Spring Framework [Електронний ресурс] – Режим доступу: https://ru.wikipedia.org/wiki/Spring_Framework
13. Spring Boot [Електронний ресурс] – Режим доступу: https://spring.io/projects/spring-boot
14. JavaScript [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/JavaScript
15. Bootstrap [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Bootstrap
16. Websocket [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/WebSocket
17. Thymeleaf [Електронний ресурс] – Режим доступу: https://www.thymeleaf.org/
18. СКБД MongoDB – [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/MongoDB

ДОДАТОК А

Тексти програмного коду

Система інформаційної підтримки навчальних курсів

(Найменування програми (документа))

DVD-R

(Вид носія даних)

27 арк, 98 Кб

(Обсяг програми (документа) , арк.,) Кб)

Київ – 2020 року

Клас CoursesApplication:

```
@SpringBootApplication
public class CoursesApplication {
        public static void main(String[] args) {
                SpringApplication.run(CoursesApplication.class, args);
        }
}
```

Клас UserController:

```
@RestController
public class UserController {
   private final UserService userService;
   private final BCryptPasswordEncoder bCryptPasswordEncoder;
   public UserController(UserService userService, BCryptPasswordEncoder bCryptPasswordEncoder){
      this.userService = userService;
      this.bCryptPasswordEncoder = bCryptPasswordEncoder;
   }
   @GetMapping("/registration")
   public ModelAndView showRegistrationPage(ModelAndView modelAndView, UserDto user) {
      modelAndView.addObject("user", user);
      modelAndView.setViewName("user/registration");
      return modelAndView;
   }
   @PostMapping("/register")
   public ModelAndView processRegistrationForm(ModelAndView modelAndView, @Valid UserDto
user, BindingResult bindingResult) {
      UserDto userExists = userService.findByEmail(user.getEmail());
      if (userExists != null) {
         modelAndView.addObject("alreadyRegisteredMessage",
               "Oops! There is already a user registered with the email provided.");
         modelAndView.setViewName("user/registration");
         bindingResult.reject("email");
      }
      if (bindingResult.hasErrors()) {
         modelAndView.setViewName("user/registration");
```

```
        } else {
            user.setPassword(bCryptPasswordEncoder.encode(user.getPassword()));
            user.setRole(Role.ROLE_STUDENT);
            userService.createUser(user);
            modelAndView.addObject("successMessage", "Hello, " + user.getFirstName()
                    + ", your account has been activated. You can login now.");
        }
        modelAndView.setViewName("user/confirmation");
        return modelAndView;
    }
}
```

Клас TeacherController:

```
@Controller
public class TeacherController {
    @Autowired
    private UserService trainerService;
    @Autowired
    private BCryptPasswordEncoder bCryptPasswordEncoder;
    private static final int ROWS_PER_PAGE = 10;
    @RequestMapping("/trainers/{page}")
    public ModelAndView allTrainers(@PathVariable("page") int page, ModelAndView model) {
        List<UserDto> trainers = trainerService.getAllByRoleAsPage(page, ROWS_PER_PAGE,
Role.ROLE_TEACHER.toString());
        model.addObject("trainers", trainers);
        model.addObject("pages", trainerService.getPagesByRole(Role.ROLE_TEACHER.toString(),
ROWS_PER_PAGE));
        model.addObject("currentUrl", "trainers");
        model.setViewName("trainer/trainers");
        return model;
    }
    @RequestMapping("/trainer-{id}")
    public ModelAndView showTrainer(@PathVariable("id") UUID id, ModelAndView model) {
        UserDto trainer= trainerService.findById(id);
        model.addObject("trainer", trainer);
        model.setViewName("trainer/show");
```

```
        return model;
    }
    @RequestMapping("/trainer-profile-{id}")
    public ModelAndView showTrainerProfile(@PathVariable("id") UUID id, ModelAndView model) {
        UserDto trainer= trainerService.findById(id);
        model.addObject("trainer", trainer);
        model.setViewName("trainer/profile");
        return model;
    }
    @GetMapping("/trainer-add")
    public ModelAndView addTrainer(ModelAndView model){
        model.addObject("user",new UserDto());
        model.setViewName("trainer/add");
        return model;
    }
    @RequestMapping(value = { "/trainer-update-{id}" }, method = RequestMethod.GET)
    public ModelAndView editTrainer(@PathVariable("id") UUID id, ModelAndView model) {
        UserDto trainer = trainerService.findById(id);
        List<Role> roles = trainerService.getRoles();
        model.addObject("roles", roles);
        Map<Long, String> mapStatus = trainerService.setMapStatus();
        model.addObject("mapStatus", mapStatus);
        trainer.setPassword(bCryptPasswordEncoder.encode(trainer.getPassword()));
        model.addObject("trainer", trainer);
        model.addObject("edit", true);
        model.setViewName("trainer/update");
        return model;
    }
    @RequestMapping(value = { "/trainer-update-{id}" }, method = RequestMethod.POST)
    public ModelAndView updateTrainer(UserDto trainer, BindingResult bindingResult, ModelAndView
model, RedirectAttributes redir){
        if(bindingResult.hasErrors())
        {
            model.setViewName("trainer/update");
            return model;
```

```
        }
        else {
          // trainerService.(trainer);
            redir.addFlashAttribute("successMessage", "User " + trainer.getFirstName() + " "+ trainer.getEmail() + " updated successfully");
            model.setViewName("redirect:/trainers/1");
            return model;
        }
    }
    @RequestMapping(value = "saveTrainer", method = RequestMethod.POST)
    public ModelAndView saveTrainer(@Valid UserDto user, BindingResult bindingResult, ModelAndView model, RedirectAttributes redir){
        UserDto userExists = trainerService.findByEmail(user.getEmail());
        if (userExists != null) {
            model.addObject("alreadyRegisteredMessage",
                    "Oops! There is already a user registered with the email provided.");
            model.setViewName("trainer/add");
            bindingResult.reject("email");
            return model;
        }
        if (bindingResult.hasErrors()) {
            model.setViewName("trainer/add");
            return model;
        }
        user.setPassword(bCryptPasswordEncoder.encode(user.getPassword()));
        user.setRole(Role.ROLE_TEACHER);
        redir.addFlashAttribute("successMessage", "User " + user.getFirstName() + " "+ user.getEmail() + " created successfully");
        model.setViewName("redirect:/trainers/1");
        return model;
    }
    @RequestMapping(value = "/trainers-delete-all", method = RequestMethod.GET)
    public ModelAndView deleteAllTrainers(ModelAndView model, RedirectAttributes redir){
        trainerService.deleteAllByRole(Role.ROLE_TEACHER.toString());
        redir.addFlashAttribute("successMessage", "All users deleted successfully");
```

```
        model.setViewName("redirect:/trainers/1");
        return model;
    }
}
```

Клас MainController:

```
@Controller
public class MainController {
    @GetMapping({"/", "/index"})
    public ModelAndView welcomePage(ModelAndView modelAndView) {
        modelAndView.setViewName("frontend/index");
        return modelAndView;
    }
    @GetMapping({"/admin"})
    public String admin(Model model) {
        return "frontend/admin";
    }
    @GetMapping("/login")
    public String loginPage(Model model) {
        return "user/login";
    }
    @GetMapping("/profilepage")
    String profilePage(Model model, Principal principal) {
        return "frontend/profilepage";
    }
    @GetMapping("/englishDescriptionPage")
    String englishDescriptionPage(Model model){
        return "frontend/englishDescriptionPage";
    }
    @GetMapping("/germanDescriptionPage")
    String germanDescriptionPage(Model model){
        return "frontend/germanDescriptionPage";
    }
    @GetMapping("/polishDescriptionPage")
    String polishDescriptionPage(Model model){
```

```
        return "frontend/polishDescriptionPage";
    }
    @GetMapping("/franceDescriptionPage")
    String franceDescriptionPage(Model model){
        return "frontend/franceDescriptionPage";
    }
    @GetMapping("/businessDescriptionPage")
    String businessDescriptionPage(Model model){
        return "frontend/businessDescriptionPage";
    }
    @GetMapping("/instructorpage")
    String instructorPage(Model model) {
        return "frontend/instructorpage";
    }
    @GetMapping("/helppage")
    String helpPage(Model model) {
        return "frontend/helppage";
    }
    @GetMapping("/website_settings")
    public ModelAndView settingsPage(ModelAndView modelAndView) {
        modelAndView.setViewName("manageSite/website_settings");
        return modelAndView;
    }
```

Клас GroupController:

```
@Controller
public class GroupController {
    @Autowired
    private GroupService groupService;
    @Autowired
    private CourseService courseService;
    @Autowired
    private UserSecurity userSecurity;
    @Autowired
    private UserService userService;
```

```
    private static final int ROWS_PER_PAGE = 10;
    @RequestMapping("/group_create/{page}/{courseId}")
    public ModelAndView showGroupListOfCourse(@PathVariable("page") int page,
                                              @PathVariable("courseId") UUID id,
                                              ModelAndView modelAndView) {
        List<GroupDto> groupList = groupService.getGroupsPage(id.toString(), page, ROWS_PER_PAGE,
                userSecurity.getLoggedInUserId().toString(), userSecurity.getLoggedInUserRole());
        CourseDto course = courseService.findById(id);
        modelAndView.addObject("courseGroup", course);
        modelAndView.addObject("pages",
                groupService.getPages(id,
                        ROWS_PER_PAGE,
                        userSecurity.getLoggedInUserId(),
                        userSecurity.getLoggedInUserRole()));
        modelAndView.addObject("id", id);
        modelAndView.addObject("groupList", groupList);
        modelAndView.addObject("currentUrl", "group_create");
        modelAndView.setViewName("groupCreator/group_create");
        return modelAndView;
    }
    @RequestMapping("/create_group_chats/{page}/{courseId}")
    public ModelAndView showGroupsWithoutChat(@PathVariable("page") int page,
                                              @PathVariable("courseId") UUID id,
                                              ModelAndView modelAndView) {
        List<GroupDto> groupList = groupService.getGroupsPageWithoutChat(id.toString(),page,ROWS_PER_PAGE);
        CourseDto course = courseService.findById(id);
        modelAndView.addObject("pages", groupService.getPagesWithoutChat(id.toString(),
ROWS_PER_PAGE));
        modelAndView.addObject("courseGroup", course);
        modelAndView.addObject("groupList", groupList);
        modelAndView.addObject("id", id);
        modelAndView.addObject("currentUrl", "create_group_chats");
        modelAndView.setViewName("chatCreator/create_group_chats");
        return modelAndView;
```

```
    }
    @GetMapping("/group_users/{page}/{groupId}")
    public ModelAndView showGroupUsersList(@PathVariable("page") int page,
                                           @PathVariable("groupId") UUID groupId,
                                           ModelAndView modelAndView) {
        List<UserDto> userList = userService.getUsersByGroupIdAsPage(page, ROWS_PER_PAGE,
groupId);
        modelAndView.addObject("pages", userService.getPagesAmountOfUsersByGroupId(groupId,
ROWS_PER_PAGE));
        modelAndView.addObject("groupId", groupId);
        modelAndView.addObject("courseId", courseService.findCourseIdByGroupId(groupId).getId());
        modelAndView.addObject("userList", userList);
        modelAndView.addObject("currentUrl", "group_users");
        modelAndView.setViewName("groupCreator/group_users");
        return modelAndView;
    }
    @GetMapping("/add_users_to_group/{groupId}")
    public ModelAndView showAddSubordinates(@PathVariable("groupId") UUID groupId,
ModelAndView model) {
        List<UserDto> users = userService.findUsersForGroupByGroupId(groupId);
        model.addObject("users", users);
        model.setViewName("groupCreator/add_users_to_group");
        return model;
    }
    @PostMapping("/add_selected_users_to_group/{groupId}")
    public ModelAndView addSelectedSubordinates(@PathVariable("groupId") UUID groupId,
                                                @RequestParam(value = "userId", required = false) UUID[] userIds,
                                                ModelAndView model, RedirectAttributes redir) {
        String message = userService.assignUsersToGroup(groupId, userIds);
        redir.addFlashAttribute("infoMessage", message);
        model.setViewName("redirect:/group_users/1/" + groupId);
        return model;
    }
    @GetMapping("/release_user_from_group/{userId}/{groupId}")
    public ModelAndView setUserFree(@PathVariable("userId") UUID userId,
```

```
                    @PathVariable("groupId") UUID groupId,
                    ModelAndView model, RedirectAttributes redir) {
        UserDto user = userService.findById(userId);
        groupService.deleteFromUserGroupByUserIdAndGroupId(userId.toString(), groupId);
        redir.addFlashAttribute("infoMessage",
            "User " + user.getFirstName() +" "+user.getLastName() + " is not a part of this group any more.");
        model.setViewName("redirect:/group_users/1/" + groupId);

        return model;
    }
    @GetMapping("/group-add/{courseId}")
    public ModelAndView addGroup(@PathVariable UUID courseId, ModelAndView modelAndView) {
        GroupDto group = new GroupDto();
        group.setCourseId(courseId.toString());
        modelAndView.addObject("group", group);
        List<CourseDto> courses = courseService.findAll();
        List<String> statuses = new ArrayList<>();
        statuses.add("Open");
        statuses.add("Closed");
        modelAndView.addObject("courses", courses);
        modelAndView.addObject("statuses", statuses);
        modelAndView.setViewName("groupCreator/group_add");
        return modelAndView;
    }
    @RequestMapping(value = "/group-save", method = RequestMethod.POST)
    public ModelAndView saveGroup(@RequestParam("courseId") String courseId, GroupDto group, ModelAndView modelAndView) {
        groupService.createGroup(group);
        modelAndView.setViewName("redirect:/group_create/1/" + courseId);
        return modelAndView;
    }
    @GetMapping({"/edit-group-{groupId}-{id}"})
    public ModelAndView editGroupBase(@PathVariable("groupId") UUID groupId,
                    @PathVariable("id") UUID id, ModelAndView modelAndView) {
```

```
        GroupDto group = groupService.findById(groupId);
        List<CourseDto> courses = courseService.findAll();
        List<String> statuses = new ArrayList<>();
        statuses.add("Open");
        statuses.add("Closed");
        modelAndView.addObject("courses", courses);
        modelAndView.addObject("statuses", statuses);
        modelAndView.addObject("group", group);
        modelAndView.addObject("edit", true);
        modelAndView.setViewName("groupCreator/edit_group_by_id");
        return modelAndView;
    }
    @PostMapping({"/edit-group-{groupId}-{id}"})
    public ModelAndView editGroupById(@PathVariable("id") UUID id, @PathVariable("groupId")
UUID groupId,
                                      GroupDto group, BindingResult bindingResult, ModelAndView modelAndView,
RedirectAttributes redirect) {
        if (bindingResult.hasErrors()) {
            modelAndView.setViewName("groupCreator/edit_group_by_id");
            return modelAndView;
        } else {
            groupService.updateGroup(groupId, group);
            modelAndView.setViewName("redirect:/group_create/1/" + id);
            return modelAndView;
        }
    }
    @GetMapping("/group-delete-by/{groupId}/{id}")
    public ModelAndView deleteGroupById(@PathVariable("groupId") UUID groupId,
                                        @PathVariable("id") Long id, ModelAndView model, RedirectAttributes
redirect) {
        groupService.deleteById(groupId);
        redirect.addFlashAttribute("successMessage", "Group deleted successfully");
        model.setViewName("redirect:/group_create/1/" + id);
        return model;
    }
```

Клас CourseController:

```
@Controller
public class CourseController {
    private final CourseService courseService;
    private final UserService userService;
    private final UserSecurity userSecurity;
    private final OrderService orderService;
    private static final int ROWS_PER_PAGE = 10;
    public CourseController(CourseService courseService, UserService userService, UserSecurity userSecurity, OrderService orderService){
        this.courseService = courseService;
        this.userSecurity = userSecurity;
        this.userService = userService;
        this.orderService = orderService;
    }
    @RequestMapping(value = "/course_create/{page}")
    public ModelAndView showCoursesList(@PathVariable("page") int page, Long courseId, ModelAndView modelAndView) {
        List<CourseDto> courseList = courseService.getCoursesPage(page, ROWS_PER_PAGE,
                userSecurity.getLoggedInUserId(), userSecurity.getLoggedInUserRole());
        System.out.println(courseList);
        modelAndView.addObject("courseList", courseList);
        modelAndView.addObject("pages",
                courseService.getPagesByUserId(userSecurity.getLoggedInUserId(),
                        ROWS_PER_PAGE,
                        userSecurity.getLoggedInUserRole()));
        modelAndView.addObject("currentUrl", "course_create");
        modelAndView.setViewName("courseCreator/course_create");
        return modelAndView;
    }
    @GetMapping("/coursepage")
    ModelAndView coursePage(ModelAndView modelAndView) {
        List<CourseDto> courseList = courseService.getCoursesPage(1, ROWS_PER_PAGE,
                userSecurity.getLoggedInUserId(), userSecurity.getLoggedInUserRole());
```

```
        System.out.println(courseList);
        modelAndView.addObject("courseList", courseList);
        modelAndView.addObject("pages",
                courseService.getPagesByUserId(userSecurity.getLoggedInUserId(),
                        ROWS_PER_PAGE,
                        userSecurity.getLoggedInUserRole()));
        modelAndView.addObject("currentUrl", "course_create");
         modelAndView.setViewName("frontend/allCourses");
        return modelAndView;
    }
    @RequestMapping(value = "/course-add", method = RequestMethod.GET)
    public ModelAndView addCourse(ModelAndView modelAndView) {
        List<UserDto> teachers = userService.findAllEnabledByRole(Role.ROLE_TEACHER.toString());
        List<String> statuses = new ArrayList<>();
        statuses.add("Активний");
        statuses.add("Закритий");
        statuses.add("Відкритий");
        modelAndView.addObject("statuses", statuses);
        modelAndView.addObject("teachers", teachers);
        modelAndView.addObject("course", new CourseDto());
        modelAndView.setViewName("courseCreator/course_add");
        return modelAndView;
    }
    @RequestMapping(value = "course-save", method = RequestMethod.POST)
    public ModelAndView saveCourse(CourseDto courseDto, ModelAndView modelAndView) {
        courseService.createCourse(courseDto);
        modelAndView.setViewName("redirect:/course_create/1");
        return modelAndView;
    }
    @RequestMapping(value = {"/edit-course-{id}"}, method = RequestMethod.GET)
    public ModelAndView editCourseBase(@PathVariable("id") UUID courseId, ModelAndView
modelAndView) {
        CourseDto course = courseService.findById(courseId);
        List<UserDto> teachers = userService.findAllEnabledByRole(Role.ROLE_TEACHER.toString());
        List<String> statuses = new ArrayList<>();
```

```
        statuses.add("Активний");
        statuses.add("Закритий");
        statuses.add("Відкритий");
        modelAndView.addObject("statuses", statuses);
        modelAndView.addObject("teachers", teachers);
        modelAndView.addObject("course", course);
        modelAndView.addObject("edit", true);
        modelAndView.setViewName("courseCreator/edit_course_by_id");
        return modelAndView;
    }
    @RequestMapping(value = {"/course-info/{courseId}"}, method = RequestMethod.GET)
    public ModelAndView getCourseInfo(@PathVariable("courseId") UUID courseId, ModelAndView modelAndView) {
        CourseDto course = courseService.findById(courseId);
        modelAndView.addObject("course", course);
        modelAndView.addObject("edit", true);
        modelAndView.setViewName("frontend/courseDetail");
        return modelAndView;
    }
    @RequestMapping(value = {"/join-course/{courseId}/{userId}"})
    public ModelAndView joinCourse(@PathVariable("courseId") UUID courseId, @PathVariable("userId") UUID userId, ModelAndView modelAndView) {
        Order order = Order.builder()
                .courseId(courseId.toString())
                .userId(userId.toString())
                .build();
        orderService.createOrder(order);
        modelAndView.setViewName("frontend/thanksPage");
        return modelAndView;
    }
    @RequestMapping(value = {"/edit-course-{id}"}, method = RequestMethod.POST)
    public ModelAndView editCourseById(@PathVariable UUID id, CourseDto courseDto, BindingResult bindingResult, ModelAndView modelAndView, RedirectAttributes redirect) {
        if (bindingResult.hasErrors()) {
            modelAndView.setViewName("courseCreator/edit_course_by_id");
```

```
            return modelAndView;
        } else {
            courseService.updateCourse(id, courseDto);
            modelAndView.setViewName("redirect:/course_create/1");
            return modelAndView;
        }
    }
    @RequestMapping(value = "/course-delete-by-{id}", method = RequestMethod.GET)
    public ModelAndView deleteCourseById(@PathVariable("id") UUID courseId, ModelAndView model, RedirectAttributes redirect) {
        courseService.deleteById(courseId);
        redirect.addFlashAttribute("successMessage", "course deleted successfully");
        model.setViewName("redirect:/course_create/1");
        return model;
    }
```

Клас LessonController:

```
@Controller
public class LessonController {
    @Autowired
    private LessonService lessonService;
    @Autowired
    private CourseService courseService;
    @Autowired
    private static final int ROWS_PER_PAGE = 10;
    @RequestMapping("/course_lessons/{page}/{courseId}")
    public ModelAndView showLessonListOfCourse(@PathVariable("page") int page,
                                               @PathVariable("courseId") UUID id,
                                               ModelAndView modelAndView) {
        List<LessonDto> lessonsOfCourse = lessonService.getLessonsByCourseId(id);
        CourseDto course = courseService.findById(id);
        modelAndView.addObject("courseLesson", course);
        modelAndView.addObject("pages", lessonService.getPages(id.toString(), ROWS_PER_PAGE));
        modelAndView.addObject("id", id);
        modelAndView.addObject("lessonsOfCourse", lessonsOfCourse);
```

```
        modelAndView.addObject("currentUrl", "course_lessons");
        modelAndView.setViewName("lessonCreator/course_lessons");
        return modelAndView;
    }
    @RequestMapping(value = "/lesson-add/{courseId}", method = RequestMethod.GET)
    public ModelAndView addLesson(@PathVariable UUID courseId, ModelAndView modelAndView) {
        LessonDto lesson = new LessonDto();
        lesson.setCourseId(courseId.toString());
        modelAndView.addObject("lesson", lesson);
        modelAndView.setViewName("lessonCreator/lesson_add");
        return modelAndView;
    }
    @RequestMapping(value = "lesson-save", method = RequestMethod.POST)
    public ModelAndView saveLesson(@RequestParam("courseId") UUID courseId,
                                   LessonDto lesson, ModelAndView modelAndView) {
        lessonService.createLesson(lesson);
        modelAndView.setViewName("redirect:/course_lessons/1/" + courseId);
        return modelAndView;
    }
    @RequestMapping(value = {"/edit-lesson/{lessonId}/{id}"}, method = RequestMethod.GET)
    public ModelAndView editLessonBase(@PathVariable("lessonId") UUID lessonId,
                                       @PathVariable("id") UUID id, ModelAndView modelAndView) {
        LessonDto lesson = lessonService.findById(lessonId);
        modelAndView.addObject("lesson", lesson);
        modelAndView.addObject("edit", true);
        modelAndView.setViewName("lessonCreator/edit_lesson_by_id");
        return modelAndView;
    }
    @RequestMapping(value = {"/edit-lesson/{lessonId}/{id}"}, method = RequestMethod.POST)
    public ModelAndView editLessonById(@PathVariable("id") UUID id, @PathVariable("lessonId")
UUID lessonId,
                                       LessonDto lesson, BindingResult bindingResult, ModelAndView
modelAndView) {
        if (bindingResult.hasErrors()) {
            modelAndView.setViewName("lessonCreator/edit_lesson_by_id");
```

```
            return modelAndView;
        } else {
            lessonService.updateLesson(lessonId, lesson);
            modelAndView.addObject("id", id);
            modelAndView.setViewName("redirect:/course_lessons/1/" + id);
            return modelAndView;
        }
    }
    @RequestMapping(value = "/lesson-delete-by/{lessonId}/{id}", method = RequestMethod.GET)
    public ModelAndView deleteLessonById(@PathVariable("lessonId") UUID lessonId,
                                         @PathVariable("id") UUID id, ModelAndView model) {
        lessonService.deleteById(lessonId);
        model.setViewName("redirect:/course_lessons/1/" + id);
        return model;
    }
}
```

Клас AttendanceController:

```
@Controller
public class AttendanceController {
    @Autowired
    private UserSecurity userSecurity;
    @Autowired
    private AttendanceService attendanceService;
    @Autowired
    private GroupService groupService;
    @Autowired
    private LessonService lessonService;
    @Autowired
    private ScheduleService scheduleService;
    @Autowired
    private CourseService courseService;
    @Autowired
    private UserService userService;
    @GetMapping(value = "/myCourses")
```

```
public ModelAndView get(ModelAndView modelAndView){
    UUID trainerId = userSecurity.getLoggedInUserId();
    UserDto trainer = userService.findById(trainerId);
    List<CourseDto> courses = courseService.findByTeacherId(trainerId);
    modelAndView.addObject("trainer", trainer);
    modelAndView.addObject("courses", courses);
    modelAndView.setViewName("attendance/main");
    return modelAndView;
}
@GetMapping(value = "myGroups/{courseId}")
public ModelAndView getGroups(
        ModelAndView modelAndView,
        @PathVariable("courseId") UUID courseId
){
    List<GroupDto> groups = groupService.findAllByCourseId(courseId.toString());
    modelAndView.addObject("groups", groups);
    modelAndView.setViewName("attendance/groups");
    return modelAndView;
}
@GetMapping(value = "myScheduleForGroup/{groupId}")
public ModelAndView getSchedules(
        ModelAndView modelAndView,
        @PathVariable("groupId") UUID groupId
){
    List<ScheduleDto> schedules = scheduleService.getSchedules(groupId.toString());
    GroupDto group = groupService.findById(groupId);
    modelAndView.addObject("schedules", schedules);
    modelAndView.addObject("group", group);
    modelAndView.setViewName("attendance/schedules");
    return modelAndView;
}
@GetMapping(value = "/presence/{groupId}/{scheduleId}")
public ModelAndView getPresence(
        ModelAndView modelAndView,
```

```
            @PathVariable("scheduleId") UUID scheduleId,
            @PathVariable("groupId") UUID groupId
    ){
        ScheduleDto schedule = scheduleService.findById(scheduleId);
        List<UserDto> users = userService.findAllByGroup(groupId);
        List<String> attendanceTypes = new ArrayList<>();
        attendanceTypes.add("Присутній");
        attendanceTypes.add("Відсутній");
        LessonDto lesson = lessonService.getLessonByScheduleId(schedule.getId().toString());
        AttendanceForm attendanceForm = attendanceService.getAttendanceListWithStudents(schedule,users);
        modelAndView.addObject("attendanceTypes",attendanceTypes);
        modelAndView.addObject("attendances", attendanceForm);
        modelAndView.addObject("lesson", lesson);
        modelAndView.addObject("users", users);
        modelAndView.addObject("schedule", schedule);
        modelAndView.setViewName("attendance/presence");
        return modelAndView;
    }
    @PostMapping(value = "/submit-attendance")
    public ModelAndView submitAttendance(
            ModelAndView model,
            @ModelAttribute AttendanceForm attendances
            ){
       attendanceService.saveAll(attendances);
        model.setViewName("redirect:/myCourses");
        return model;
    }
    @GetMapping(value = "mySubordinates")
    public ModelAndView getSubordinatesAttendance(
            ModelAndView modelAndView
    ){
        UUID managerId = userSecurity.getLoggedInUserId();
        // Find all subordinates of the manager by manager's id
```

```
        List<AttendanceDto> attendances = attendanceService.getSubordinatesAttendanceByManager(managerId);
        modelAndView.addObject("attendances", attendances);
        modelAndView.setViewName("attendance/subordinatesAttendances");
        return modelAndView;
    }
}
```

Клас CourseSeviceImpl:

```
@Service
public class CourseServiceImpl implements CourseService {
    private final CourseRepository courseRepository;
    private final UserService userService;
    private final GroupRepository groupRepository;
    private final UserGroupRepository userGroupRepository;
    public CourseServiceImpl(CourseRepository courseRepository, UserService userService,
                             GroupRepository groupRepository, UserGroupRepository userGroupRepository) {
        this.courseRepository = courseRepository;
        this.userService = userService;
        this.groupRepository = groupRepository;
        this.userGroupRepository = userGroupRepository;
    }
    @Override
    public CourseDto createCourse(CourseDto courseDto) {
        CourseEntity createdCourse = courseRepository.save(toEntity(courseDto));
        return toDto(createdCourse);
    }
    @Override
    public CourseDto updateCourse(UUID courseId, CourseDto courseDto) {
        CourseEntity updated = courseRepository.save(toEntity(courseDto));
        return toDto(updated);
    }
    @Override
    public void deleteById(UUID id) {
        courseRepository.deleteById(id);
```

```
    }
    @Override
    public CourseDto findById(UUID courseId) {
        Optional<CourseEntity> courseEntityOptional = courseRepository.findById(courseId);
        return toDto(courseEntityOptional.get());
    }
    @Override
    public List<CourseDto> findAll() {
        List<CourseDto> courses = courseRepository.findAll().stream().map(this::toDto).collect(Collectors.toList());
        for (CourseDto course: courses
            ) {
            course.setTeacher(userService.findById(UUID.fromString(course.getTeacherId())));
        }
        return courses;
    }
    @Override
    public int getPages(double rowsPerPage) {
        return 0;
    }
    @Override
    public List<CourseDto> findByTeacherId(UUID id) {
        return courseRepository.findByTeacherId(id.toString()).stream().map(this::toDto).collect(Collectors.toList());
    }


    @Override
    public int getPagesByUserId(UUID userId, double rowsPerPage, String role) {
        return 0;
    }
    @Override
    public List<CourseDto> getCoursesPage(int page, int rowsPerPage, UUID userId, String role) {
        List<CourseDto> courseList;


        if (role.equals("ROLE_ADMIN")) {
```

```
            courseList = courseRepository.findAll().stream().map(this::toDto).collect(Collectors.toList());
        } else if (role.equals("ROLE_TEACHER")) {
            courseList = courseRepository.findAllByTeacherId(userId.toString()).stream().map(this::toDto).collect(Collectors.toList());
        } else {
            //courseList = courseRepository.getAllAsPageByEmployeeId(userId, page, rowsPerPage);
            courseList = courseRepository.findAll().stream().map(this::toDto).collect(Collectors.toList());
        }
        for (CourseDto course : courseList) {
            course.setTeacher(userService.findById(UUID.fromString(course.getTeacherId())));
        }
        return courseList;
    }
    @Override
    public List<CourseDto> findCoursesByUser(UserDto userDto) {
        List<CourseDto> courseList;

        if(userDto.getRole().toString().equals(Role.ROLE_TEACHER.toString())) {
            courseList = courseRepository.findByTeacherId(userDto.getId().toString()).stream().map(this::toDto).collect(Collectors.toList());
        } else {
            List<UserGroup> userGroups = userGroupRepository.findAllByUserId(userDto.getId().toString());
            List<String> groupIds = userGroups.stream().map(UserGroup::getGroupId).filter(Objects::nonNull).collect(Collectors.toList());
            List<GroupEntity> groupEntities = new ArrayList<>();
            for (String groupId: groupIds
                 ) {
                Optional<GroupEntity> groupEntityOptional = groupRepository.findById(UUID.fromString(groupId));
                if (groupEntityOptional.isPresent()) {
                    GroupEntity groupEntity = groupEntityOptional.get();
                    groupEntities.add(groupEntity);
                }
            }
```

```
        List<String> courseIds = groupEntities.stream().map(GroupEntity::getCourseId).filter(Objects::nonNull).collect(Collectors.toList());
        List<CourseEntity> courseEntities = new ArrayList<>();
        for (String courseId: courseIds
             ) {
            Optional<CourseEntity> courseEntityOptional = courseRepository.findById(UUID.fromString(courseId));
            if(courseEntityOptional.isPresent()) {
                CourseEntity courseEntity = courseEntityOptional.get();
                courseEntities.add(courseEntity);
            }
        }
        courseList = courseEntities.stream().map(this::toDto).filter(Objects::nonNull).collect(Collectors.toList());
        for(CourseDto course : courseList){
            course.setTeacher(userService.findById(UUID.fromString(course.getTeacherId())));
        }
    }
    return courseList;
}
@Override
public CourseDto findCourseIdByGroupId(UUID groupId) {
    Optional<GroupEntity> groupEntityOptional = groupRepository.findById(groupId);
    if(groupEntityOptional.isPresent()) {
        GroupEntity groupEntity = groupEntityOptional.get();

        CourseEntity courseEntity = courseRepository.findById(UUID.fromString(groupEntity.getCourseId())).get();
        return toDto(courseEntity);
    }
    else return null;
}
@Override
public void upsert(CourseDto courseDto) {
    if (StringUtils.isEmpty(courseDto.getId())) {
```

```
            throw new IllegalArgumentException("Id is not define");
        }
        courseRepository.upsert(toEntity(courseDto));
    }
    CourseDto toDto(CourseEntity courseEntity){
        if(courseEntity == null) return null;
      return CourseDto.builder()
                .id(courseEntity.getId())
                .name(courseEntity.getName())
                .description(courseEntity.getDescription())
                .courseStatus(courseEntity.getCourseStatus())
                .teacher(courseEntity.getTeacher())
                .startDate(courseEntity.getStartDate())
                .endDate(courseEntity.getEndDate())
                .teacherId(courseEntity.getTeacherId())
                .filename(courseEntity.getFilename())
                .build();
    }
    CourseEntity toEntity(CourseDto courseDto){
        if(courseDto == null) return null;
        UUID id = UUID.randomUUID();
        if (courseDto.getId()==null) courseDto.setId(id);
        return CourseEntity.builder()
                .id(courseDto.getId())
                .name(courseDto.getName())
                .description(courseDto.getDescription())
                .courseStatus(courseDto.getCourseStatus())
                .teacher(courseDto.getTeacher())
                .startDate(courseDto.getStartDate())
                .endDate(courseDto.getEndDate())
                .teacherId(courseDto.getTeacherId())
                .filename(courseDto.getFilename())
                .build();
    }
```

Клас ApplicationStartup:

```
@Component
public class ApplicationStartup implements ApplicationListener<ApplicationStartedEvent>, Ordered {
    private final ConfigLoaderService configLoaderService;
    public ApplicationStartup(ConfigLoaderService configLoaderService) {
        this.configLoaderService = configLoaderService;
    }
    @Override
    public void onApplicationEvent(final ApplicationStartedEvent applicationStartedEvent) {
        configLoaderService.loadAttendanceTypes("defaultConfig/attendanceTypeConfig.json");
        configLoaderService.loadDefaultUsers("defaultConfig/defaultUsersConfig.json");
        configLoaderService.loadDefaultCourses("defaultConfig/defaultCoursesConfig.json");
    }
    @Override
    public int getOrder() {
        return 10;
    }
}
```

Клас ConfigLoaderService:

```
@Component
@Slf4j
public class ConfigLoaderService {
    private final ResourceLoader resourceLoader;
    private final ObjectMapper objectMapper;
    private final AttendanceTypeService attendanceTypeService;
    private final UserService userService;
    private final CourseService courseService;
    public ConfigLoaderService(@Qualifier("webApplicationContext")ResourceLoader resourceLoader,
                               UserService userService,
                               AttendanceTypeService attendanceTypeService,
                               ObjectMapper objectMapper,
                               CourseService courseService) {
        this.resourceLoader = resourceLoader;
        this.objectMapper = objectMapper;
```

```
        this.attendanceTypeService = attendanceTypeService;
        this.userService = userService;
        this.courseService = courseService;
    }
    public List<AttendanceType> loadAttendanceTypes(String fileName) {
        final List<AttendanceType> types = readConfig(fileName, new
TypeReference<List<AttendanceType>>() {
        });
        types.forEach(attendanceTypeService::upsert);
        return types;
    }
    public List<UserDto> loadDefaultUsers(String fileName) {
        final List<UserDto> users = readConfig(fileName, new TypeReference<List<UserDto>>() {
        });
        users.forEach(userService::upsert);
        return users;
    }
    public List<CourseDto> loadDefaultCourses(String fileName) {
        final List<CourseDto> courses = readConfig(fileName, new TypeReference<List<CourseDto>>() {
        });
        courses.forEach(courseService::upsert);
        return courses;
    }
    private <T> T readConfig(String fileName, TypeReference<T> reference) {
        try {
            Resource resource = resourceLoader.getResource("classpath:" + fileName);
            BufferedReader reader = new BufferedReader(new
InputStreamReader(resource.getInputStream()));
            return objectMapper.readValue(reader, reference);
        } catch (IOException e) {
            throw new RuntimeException(String.format("Error while loading default queues, file: %s",
fileName), e);
        }
    }
```

Клас ProfileServiceImpl:

```
@Service
public class ProfileServiceImpl implements ProfileService {

    private final UserService userService;

    public ProfileServiceImpl(UserService userService) {
        this.userService = userService;
    }

    @Override
    public UserDto initManager(UserDto user) {

        if (user.getRole().toString().equals(Role.ROLE_STUDENT.toString()) && user.getManagerId() != null) {
            return userService.findManagerBySubordinateId(user.getId());
        } else {
            return new UserDto();
        }
    }
    @Override
    public String getReadableRole(UserDto user) {
        String role;
        String roleId = user.getRole().toString();

        if (roleId.equals(Role.ROLE_ADMIN.toString())) {
            role = "Admin";
        } else if (roleId.equals(Role.ROLE_MANAGER.toString())) {
            role = "Manager";
        } else if (roleId.equals(Role.ROLE_TEACHER.toString())) {
            role = "Teacher";
        } else {
            role = "Student";
        }
```

```
        return role;
    }
}
```

Клас UserDetailsServiceImpl:

```
@Service
public class UserDetailsServiceImpl implements UserDetailsService {
  private final UserService userService;
   public UserDetailsServiceImpl(UserService userService){
   this.userService = userService;
   }
   @Override
   public UserDetails loadUserByUsername(String email) throws UsernameNotFoundException {
      UserDto existingUser = userService.findByEmail(email);
      if (existingUser == null) {
         throw new UsernameNotFoundException(email + " was not found in the database");
      }
      List<String> roleNames = new ArrayList<>();
      roleNames.add(existingUser.getRole().toString());
      List<GrantedAuthority> grantList = new ArrayList<GrantedAuthority>();
      System.out.println("roles: "+roleNames.toString());
      if (roleNames != null) {
         for (String role : roleNames) {
            GrantedAuthority authority = new SimpleGrantedAuthority(role);
            grantList.add(authority);
         }
      }
      return (UserDetails) new LoggedInUser(existingUser.getEmail(),
            existingUser.getPassword(), grantList, existingUser.getId());
   }
}
```

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”
Кафедра автоматизованих систем обробки інформації і управління

УЗГОДЖЕНО	**ЗАТВЕРДЖУЮ**
Керівник проєкту	**В.о. завідувача кафедри**
_____ *Олесь КОВТУНЕЦЬ*	_____ *Олександр ПАВЛОВ*
(підпис) (вл. ім’я, прізвище)	(підпис) (вл. ім’я, прізвище)
“13” квітня 2020 р.	“14” квітня 2020 р.

Система інформаційної підтримки навчальних курсів

ТЕХНІЧНЕ ЗАВДАННЯ

Шифр *ДП 6213.01.000 ТЗ*

На 9 сторінках

Київ – 2020 року

ЗМІСТ

1 ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1 Повне найменування системи та її умовне позначення

Повне найменування системи: «Система інформаційної підтримки навчальних курсів».

Умовне позначення: «Система навчальних курсів».

Скорочена назва: «Система».

1.2 Найменування організації-замовника та організацій-учасників робіт

Організація-замовник: ТОВ «НЕТКРЕКЕР».

Виконавець: студент групи ІС-62, кафедра АСОІУ ФІОТ КПІ ім. Ігоря Сікорського, Мусієнко Віталій Сергійович.

1.3 Перелік документів, на підставі яких створюється система

Підставою для розробки «Системи інформаційної підтримки навчальних курсів» є завдання на дипломне проектування, затверджене кафедрою автоматизованих систем обробки інформації та управління Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

1.4 Планові терміни початку і закінчення роботи зі створення системи

Плановий термін початку виконання робіт: 1 березня 2020 року.

Плановий термін закінчення виконання робіт: 31 травня 2020 року.

2 ПРИЗНАЧЕННЯ І ЦІЛІ СТВОРЕННЯ СИСТЕМИ

2.1 Призначення системи

Призначенням даної розробки є допомога в організації навчального процесу на курсах. Система повинна давати можливість студентам, викладачам та менеджерам зібрати всю необхідну інформацію в одному місці.

2.2 Цілі створення системи

Цілі створення системи:

- Допомога в організації навчального процесу на курсах;
- Можливість швидко і зручно знаходити курси по власних інтересах;
- Покращити ефективність навчання студентів.

3 ХАРАКТЕРИСТИКА ОБ'ЄКТА АВТОМАТИЗАЦІЇ

Об’єктом автоматизції є процес навчання на курсах.

Для реалізації поставленої задачі необхідно:

1) Зібрати усю необхідну інформацію в одному місці та надати до неї доступ усім учасникам навчального процесу;
2) Полегшити процес пошуку навчальних курсів шляхом введення системи рекомендацій;
3) Розробити зручний спосіб комунікації учасників навчального процесу всередині системи.

Для полегшення пошуку навчальних курсів необхідно розробити систему рекомендацій за застосувати її в нашій системі. Одним із методів побудови прогнозів в рекомендаційних системах є метод колаборативної фільтрації. Один із варіантів його реалізції, а саме – метод колаборативної фільтрації на основі подібності елементів, буде використаний у даній роботі для побудови системи рекомендацій.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

Програмне забезпечення повинно забезпечувати виконання наступних основних функцій:

Для користувача:

1) Реєстрація в системі;
2) Вхід у систему;
3) Перегляд навчальних курсів;
4) Перегляд розкладу занять;
5) Перегляд профілю свого менеджера та викладача курсу;
6) Можливість писати повідомлення іншим учасникам системи.

Для адміністратора:

1) Створення курсу;
2) Створення групи;
3) Створення групових чатів;
4) Розподілення студентів по групах;
5) Створення, редагування та видалення профілів для менеджерів;
6) Створення, редагування та видалення профілів для викладачів;
7) Перегляд профілю всіх користувачів системи;
8) Створення розкладу для груп.

Для викладача:

1) Перевірка відвідуваності студентів;
2) Створення розкладу занять;
3) Перегляд профілю своїх студентів.

Для менеджера:

1) Перегляд списку своїх студентів;
2) Перегляд профілю своїх студентів.

4.2 Вимоги до надійності

Висуваються наступні вимоги до надійності програмного забезпечення:

- передбачити контроль введення інформації
- передбачити захист від некоректних дій користувача
- обмежити користувачам доступ до певних сторінок, відповідно до їхньої ролі

4.3 Умови експлуатації

Для коректної роботи системи необхідний пристрій з платформою, яка відповідає вимогам зазначеним в розділі 4.4 .

4.4 Вимоги до складу і параметрів технічних засобів

Мінімальна конфігурація технічних засобів

- Тип процесору: Intel Core i3+;
- Об’єм ОЗП: 4 ГБ;
- Доступ до інтернету;
- Наявність веб-браузера.

5 СТАДІЇ ТА ЕТАПИ РОЗРОБКИ

У таблиці 5.1 наведено календарний план робіт та терміни їх виконання.

Таблиця 5.1 – Календарний план виконання робіт

№ з/п	Назва етапів створення продукту	Строк виконання
1.	Вивчення рекомендованої літератури	10.03.2020 р.
2.	Аналіз існуючих методів розв’язання задачі	15.03.2020 р.
3.	Постановка та формалізація задачі	28.03.2020 р.
4.	Розробка інформаційного забезпечення	7.05.2020 р.
5.	Алгоритмізація задачі	22.03.2020 р.
6.	Обґрунтування використовуваних технічних засобів	30.03.2020 р.
7.	Розробка програмного забезпечення	10.04.2020 р.
8.	Налагодження програми	17.04.2020 р.
9.	Виконання графічних документів	23.04.2020 р.
10.	Оформлення пояснювальної записки	01.05.2020 р.
11.	Подання ДП на попередній захист	15.05.2020 р.
12.	Подання ДП на основний захист	01.06.2020 р.
13.	Подання ДП рецензенту	02.06.2020 р.

6 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

6.1 Види випробувань

Для перевірки правильності роботи системи буде проведено функціональне тестування. В ході тестування буде виконано перевірку всіх функціональних характеристик веб-застосунку. Буде виконана перевірка на відмовостійкість шляхом введення некоректних даних користувачем. Окремими тестами буде перевірена загальна безпека системи.

UNICHECK

Власник документу:
Попенко Володимир Дмитрович

ID перевірки:
1003932318

Дата перевірки:
10.06.2020 14:32:57 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
10.06.2020 14:34:01 EEST

ID користувача:
77149

Назва документу: Musienko_bachelor_is62_2

ID файлу: 1003947692 Кількість сторінок: 61 Кількість слів: 7793 Кількість символів: 57615 Розмір файлу: 1.74 MB

14.9% Схожість

Найбільша схожість: 4.97% з джерело бібліотеки. ID файлу: 5850093

6.99% Схожість з Інтернет джерелами	106	Page 63
14.5% Текстові збіги по Бібліотеці акаунту	481	Page 64

0% Цитат

Не знайдено жодних цитат

0% Вилучень

Вилучений текст відсутній

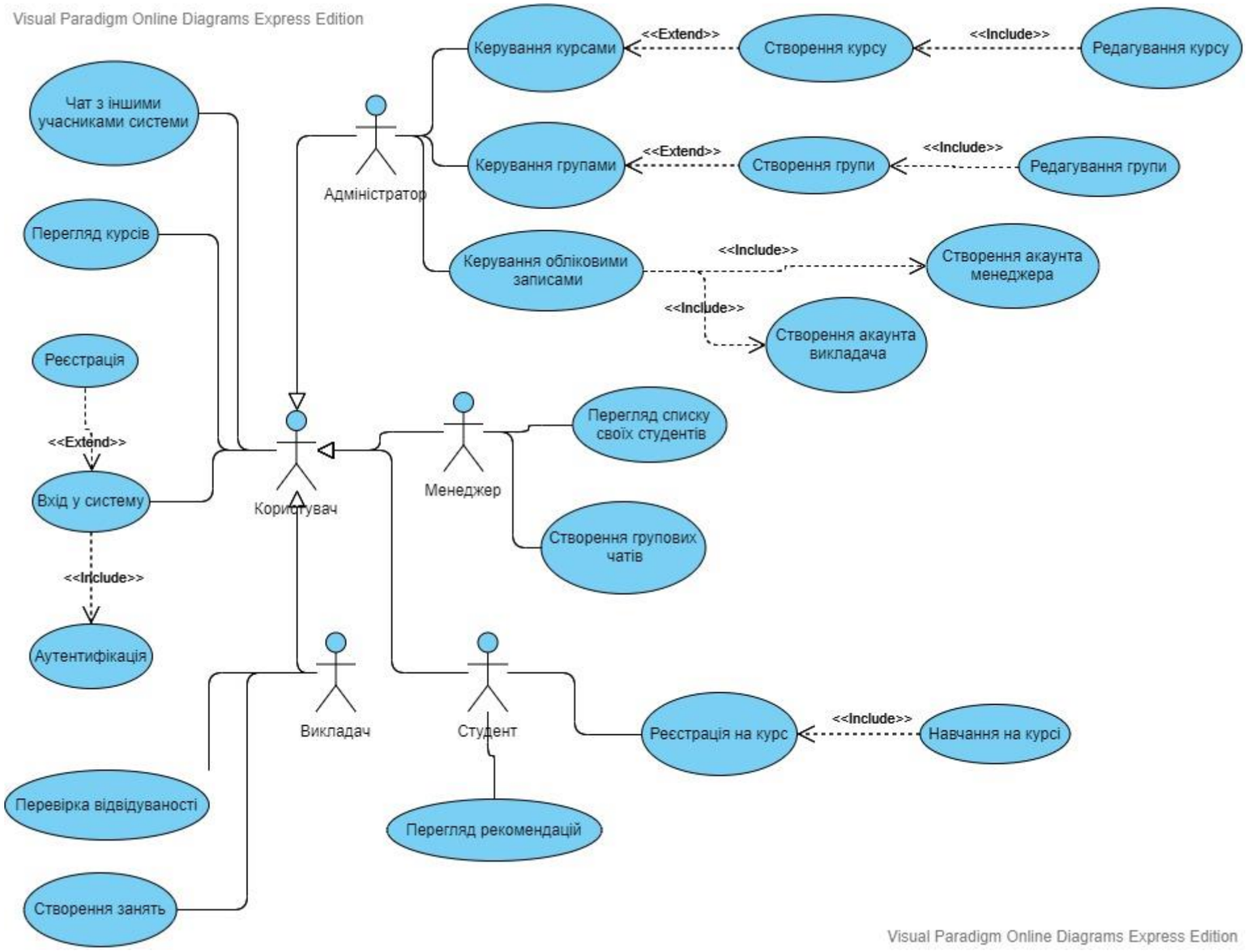
Підміна символів

Не знайдено замінених символів

Графічний матеріал
до дипломного проєкту

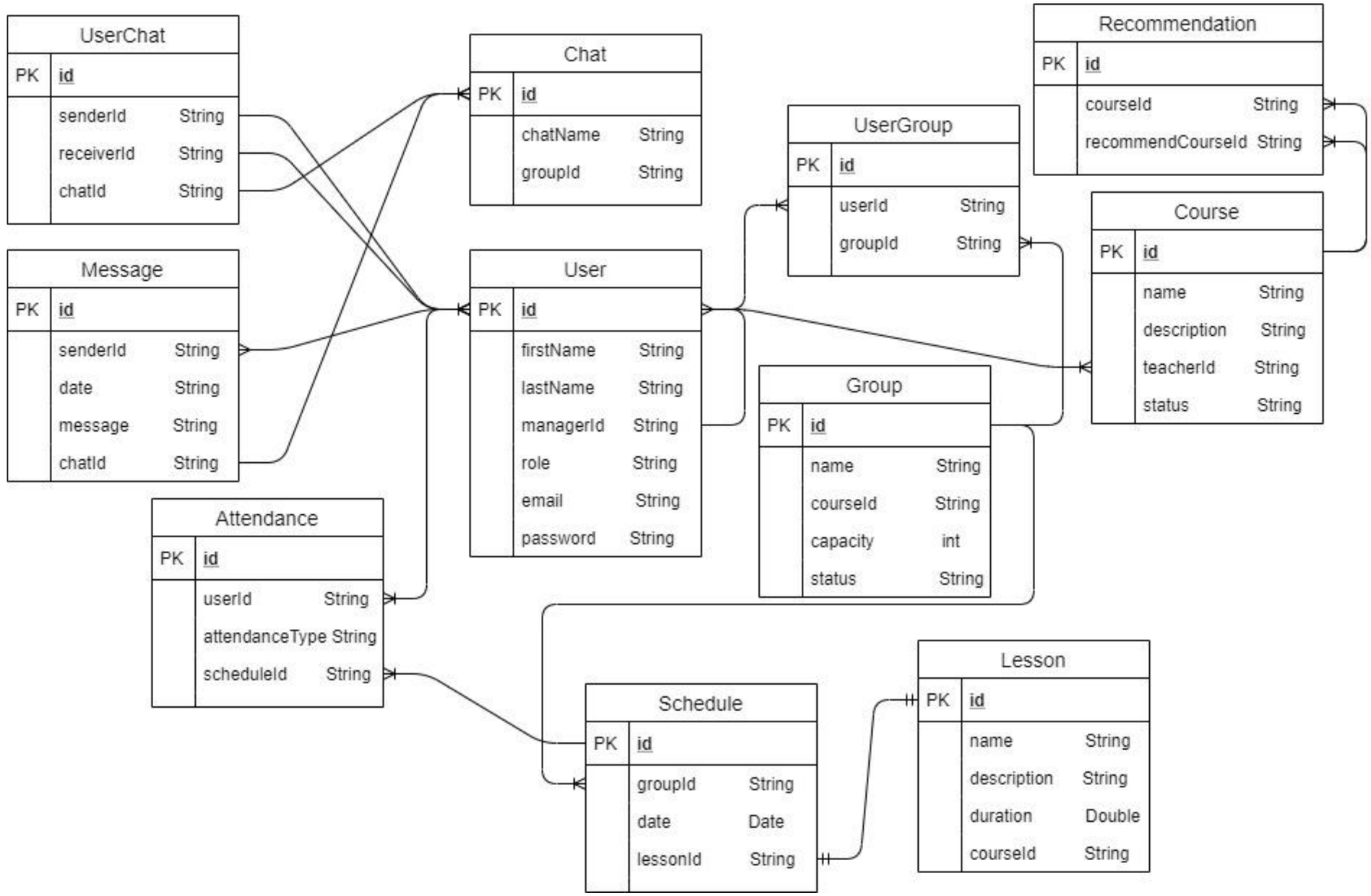
на тему: Система інформаційної підтримки навчальних курсів

Київ – 2020 року

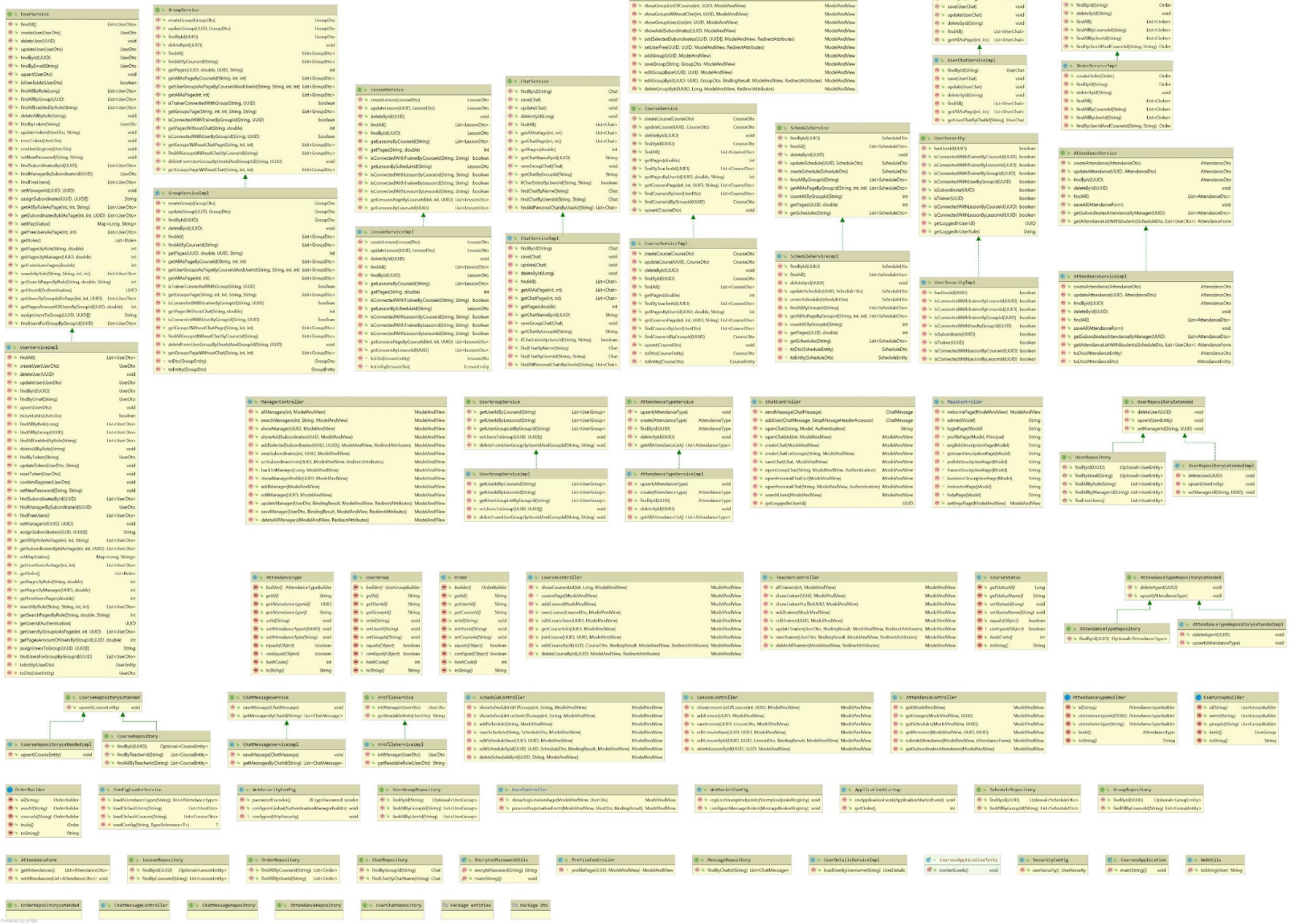


					ДП 6213.02.000 ССВ	
					Схема структурна варіантів використань	Літера: — | Маса: — | Масштаб: —
Зм.	Арк.	№ документа	Підпис	Дата		Аркуш 1 | Аркушів 1
Розробив		Мусієнко В.С.				
Перевірив		Ковтунець О.В.				
Консульт.					*Система інформаційної підтримки навчальних курсів*	*КПІ ім. Ігоря Сікорського*
Н. кон.		Новінський В.П.				*Каф. АСОІУ*
Затвердив		Ковтунець О.В.				*Гр. ІС-62*

ДП 6213.03.000 СБД



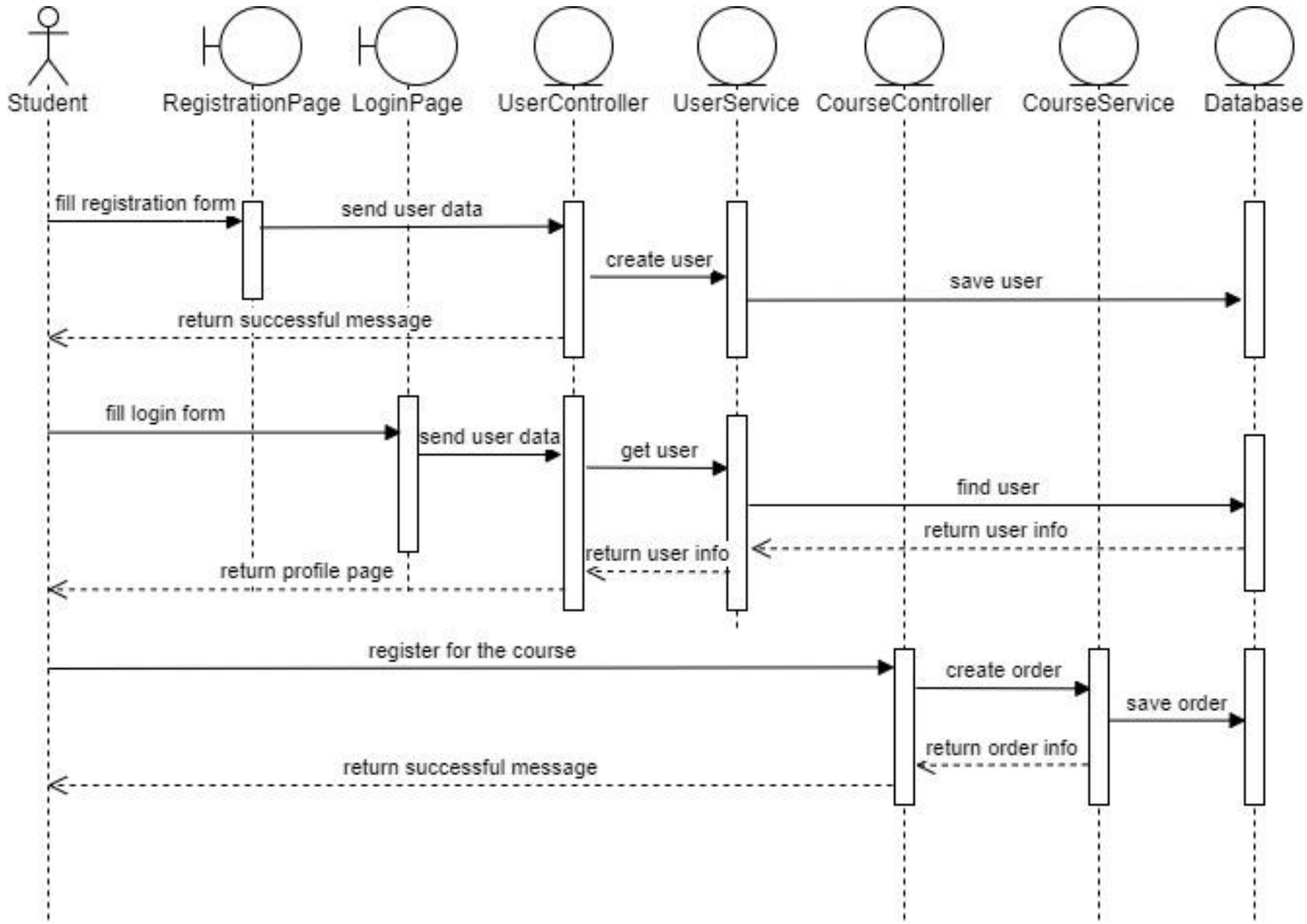
					ДП 6213.03.000 СБД	
Зм.	Арк.	№ документа	Підпис	Дата	Схема бази даних	Літера / Маса / Масштаб
Розробив		Мусієнко В.С.				
Перевірив		Ковтунець О.В.				Аркуш 1 / Аркушів 1
Консульт.					Система інформаційної підтримки навчальних курсів	КПІ ім. Ігоря Сікорського Каф. АСОІУ Гр. ІС-62
Н. кон.		Новінський В.П.				
Затвердив		Ковтунець О.В.				



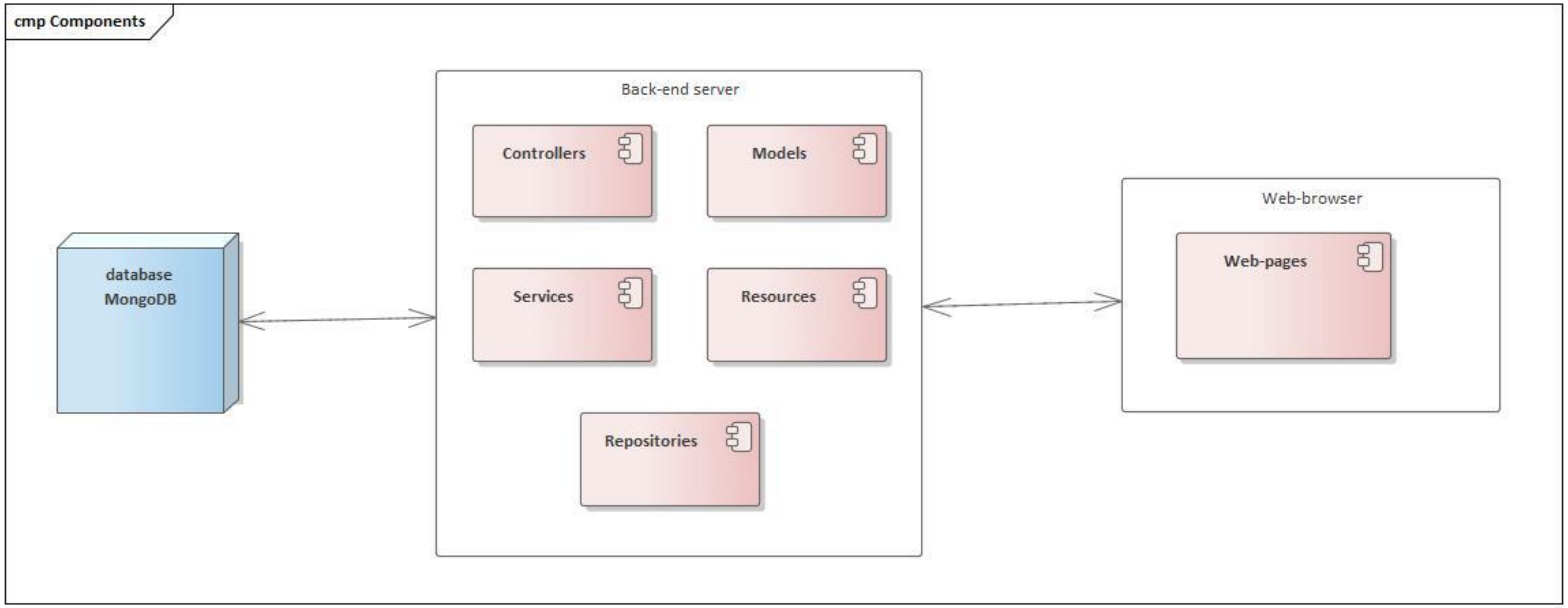
						ДП 6213.04.000 ССК
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна класів програмного забезпечення	Літера / Маса / Масштаб
Розробив		Мусієнко В.С.				
Перевірив		Ковтунець О.В.				Аркуш 1 / Аркушів 1
Консульт.					Система інформаційної підтримки навчальних курсів	КПІ ім. Ігоря Сікорського Каф. АСОІУ Гр. ІС-62
Н. кон.		Новінський В.П.				
Затвердив		Ковтунець О.В.				

ДП 6213.04.000 ССК

ДП 6213.05.000 ССП



					ДП 6213.05.000 ССП			
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна послідовності	Літера	Маса	Масштаб
Розробив		Мусієнко В.С.						
Перевірив		Ковтунець О.В.				Аркуш 1	Аркушів 1	
Консульт.					Система інформаційної підтримки навчальних курсів	КПІ ім. Ігоря Сікорського Каф. АСОІУ Гр. ІС-62		
Н. кон.		Новінський В.П.						
Затвердив		Ковтунець О.В.						

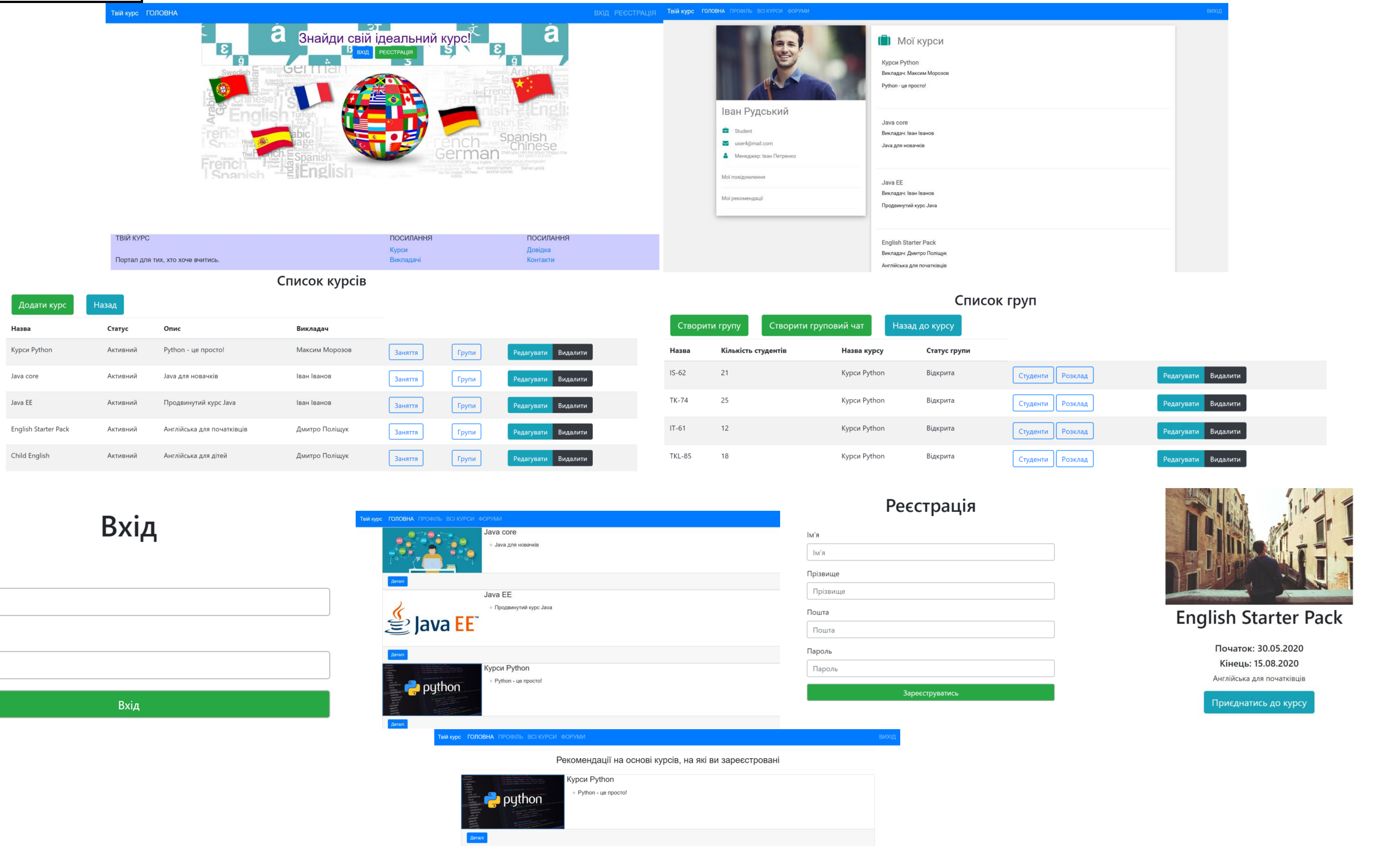


					ДП 6213.06.000 ССК
Зм.	Арк.	№ документа	Підпис	Дата	*Схема структурна компонентів програмного забезпечення*
Розробив		Мусієнко В.С.			
Перевірив		Ковтунець О.В.			
Консульт.					*Система інформаційної підтримки навчальних курсів*
Н. кон.		Новінський В.П.			
Затвердив		Ковтунець О.В.			

Літера	Маса	Масштаб
Аркуш 1	Аркушів 1	

КПІ ім. Ігоря Сікорського
Каф. АСОІУ
Гр. ІС-62

ДП 6213.07.000 ЕК



					ДП 6213.07.000 ЕК			
Зм.	Арк.	№ документа	Підпис	Дата	Креслення вигляду екранних форм	Літера	Маса	Масштаб
Розробив		Мусієнко В.С.						
Перевірив		Ковтунець О.В.				Аркуш 1	Аркушів 1	
Консульт.					Система інформаційної підтримки навчальних курсів	КПІ ім. Ігоря Сікорського Каф. АСОІУ Гр. ІС-62		
Н. кон.		Новінський В.П.						
Затвердив		Ковтунець О.В.						